Algoritma dan Pemrograman R

Bakti Siregar, M.Sc

20 Agustus 2022

Contents

Kata Pengantar

Bahasa pemrograman R telah menjadi alat yang kuat bagi para ilmuwan data, analis statistik, dan praktisi analisis numerik di seluruh dunia. Dengan kemampuan yang luar biasa dalam manipulasi data, visualisasi, dan analisis statistik, R memungkinkan para profesional untuk menggali wawasan berharga dari kumpulan data yang kompleks.

Dalam buku ini, penulis menyediakan materi dasar-dasar bahasa pemrograman R hingga tingkat yang lebih mendalam. Penulis juga menjelaskan beberapa konsep-konsep penting, sintaksis dasar, struktur data, serta memberikan contoh nyata tentang bagaimana R dapat digunakan dalam berbagai konteks. Modul ini dirancang untuk membantu pembaca yang baru mengenal pemrograman maupun yang telah memiliki pengalaman sebelumnya dalam bahasa lain.

Ringkasan Pembelajaran

Adapun isi pembelajaran dalam modul ini adalah sebagai berikut:

- **Minggu 1-2: Dasar Pemrograman R**
  - Pengenalan konsep dasar pemrograman.
  - Pengenalan lingkungan dan pengaturan awal bahasa R.
  - Menulis program sederhana dalam R.
- **Minggu 3-4: Variabel dan Tipe Data**
  - Mengenal tipe data dasar dalam R: numerik, karakter, logika.
  - Deklarasi dan penggunaan variabel.
  - Konversi antar tipe data.
- **Minggu 5-6: Struktur Kontrol**
  - Penggunaan pernyataan kondisional (if, else).
  - Penggunaan perulangan (for, while) untuk mengatur alur program.
  - Studi kasus penggunaan struktur kontrol dalam pemrograman.
- **Minggu 7-8: Fungsi**

  - Pengenalan konsep fungsi dalam pemrograman.
  - Membuat dan memanggil fungsi dalam R.
  - Penggunaan parameter dalam fungsi.

- **Minggu 9-10: Struktur Data**
  - Pengenalan array dan matriks dalam R.
  - Penggunaan vektor dan faktor.
  - Pengenalan konsep dataframe untuk manipulasi data tabular.

- **Minggu 11-12: Algoritma Dasar**
  - Pengenalan konsep algoritma dan kompleksitas.
  - Pemahaman tentang pencarian dan pengurutan.
  - Implementasi algoritma pencarian dan pengurutan sederhana dalam R.

- **Minggu 13-14: Pengenalan Analisis Data**
  - Pengenalan pustaka dasar untuk analisis data di R.
  - Pemahaman tentang statistik dasar dan visualisasi data.
  - Penerapan analisis sederhana pada dataset kecil.

- **Minggu 15-16: Proyek Akhir** Siswa diminta untuk membuat proyek kecil menggunakan R yang menggabungkan konsep-konsep yang telah dipelajari. Proyek dapat berupa analisis data, pemecahan masalah, atau aplikasi sederhana.

Tim Penyusun

Berikut ini adalah nama dan biografi singkat para penulis:

- **Bakti Siregar, M.Sc** adalah Ketua Program Studi di Jurusan Statistika Universitas Matana. Lulusan Magister Matematika Terapan dari National Sun Yat Sen University, Taiwan. Beliau juga merupakan dosen dan konsultan Data Scientist di perusahaan-perusahaan ternama seperti JNE, Samora Group, Pertamina, dan lainnya. Beliau memiliki antusiasme khusus dalam mengajar Big Data Analytics, Machine Learning, Optimisasi, dan Analisis Time Series di bidang keuangan dan investasi. Keahliannya juga terlihat dalam penggunaan bahasa pemrograman Statistik seperti R Studio dan Python. Beliau mengaplikasikan sistem basis data MySQL/NoSQL dalam pembelajaran manajemen data, serta mahir dalam menggunakan tools Big Data seperti Spark dan Hadoop. Beberapa project beliau dapat dilihat di link berikut: Rpubs, Github, Website, dan Kaggle.

- **Yonathan Anggraiwan, S.Stat** adalah seorang mahasiswa yang bersemangat dalam dunia pemrograman dan analisis data. Lahir di Tangerang,

minatnya terhadap teknologi dan komputer muncul sejak usia dini. Ia tumbuh dengan rasa ingin tahu yang kuat terhadap bahasa pemrograman, dan ini membawanya menuju dunia analisis data menggunakan bahasa pemrograman R dan Python. Selama menjalankan tugas sebagai asisten lab, Yonathan Anggraiwan berperan dalam membantu mahasiswa dalam memahami konsep-konsep dasar dan kompleks dalam pemrograman R dan Python. Ia memberikan penjelasan yang jelas dan dukungan kepada mahasiswa yang mengalami kesulitan. Selain itu, ia juga terlibat dalam merancang tugas dan ujian praktikum, serta memberikan umpan balik konstruktif kepada para mahasiswa. Dalam perjalanan waktu, Yonathan Anggraiwan mulai mengambil tanggung jawab lebih besar dalam laboratorium. Ia membantu mengembangkan materi pembelajaran tambahan, seperti tutorial online tentang analisis data menggunakan R dan Python. Ia juga aktif dalam berbagai proyek penelitian di bawah bimbingan dosen, yang melibatkan pengolahan data besar untuk analisis statistik dan visualisasi. Dengan semangat yang tinggi, dedikasi, dan keterampilan yang dimilikinya, Yonathan Anggraiwan Ridwan adalah contoh nyata dari seorang mahasiswa yang berhasil menggabungkan minatnya dalam pemrograman R dan Python dengan peran yang produktif sebagai asisten laboratorium dan kontributor dalam dunia analisis data.

Ucapan Terima Kasih

Kami berharap modul ini akan menjadi panduan yang bermanfaat bagi Anda dalam menguasai bahasa pemrograman R. Semoga dengan memahami konsep-konsep yang disajikan dalam modul ini, Anda akan dapat mengaplikasikan R dalam proyek-proyek analisis data dan statistik yang sebenarnya.

Terima kasih kepada semua yang telah berkontribusi dalam pembuatan modul ini, serta kepada Anda, pembaca, yang telah memilih modul ini sebagai sumber pengetahuan Anda. Kami berharap Anda menikmati perjalanan Anda dalam memahami bahasa pemrograman R.

Masukan & Saran

Semua masukan dan tanggapan Anda sangat berarti bagi kami untuk memperbaiki modul ini kedepannya. Bagi para pembaca/pengguna yang ingin menyampaikan masukan dan tanggapan, dipersilahkan melalui kontak dibawak ini!

Email: dsciencelabs@outlook.com

Chapter 1

Pengenalan R?

R adalah bahasa pemrograman dan lingkungan komputasi yang digunakan untuk analisis statistik, visualisasi data, pengolahan data, dan pemodelan prediktif. R dikembangkan oleh Ross Ihaka dan Robert Gentleman di Universitas Auckland, Selandia Baru. R menjadi populer dalam dunia analisis data dan ilmu data karena kemampuannya dalam mengolah dan menganalisis data secara efisien.

1.1 Fitur Utama R

1. **Open Source:** R adalah perangkat lunak open source yang dapat diunduh dan digunakan secara gratis.
2. **Fleksibilitas:** Anda dapat membuat fungsi sendiri, mengontrol alur program, dan mengakses berbagai pustaka eksternal.
3. **Mengimpor dan Mengekspor Data:** R mendukung berbagai format file, seperti CSV, Excel, SQL, dan format data lainnya.
4. **Data Manipulasi:** R memiliki pustaka seperti dplyr dan tidyr yang memudahkan manipulasi dan transformasi data.
5. **Lingkungan Komputasi:** R tidak hanya bahasa pemrograman, tetapi juga lingkungan komputasi lengkap yang menyediakan alat untuk analisis dan visualisasi data.
6. **Statistik dan Analisis Data:** R memiliki beragam pustaka dan paket yang mendukung analisis statistik, visualisasi data, dan pemodelan prediktif.
7. **Grafik dan Visualisasi:** R memiliki kemampuan visualisasi yang kuat dengan pustaka seperti ggplot2 untuk membuat grafik yang informatif dan menarik.
8. **Komunitas Aktif:** Komunitas R sangat aktif, dan ada banyak sumber

daya online, forum, dan pustaka yang dapat membantu dalam pembelajaran dan pemecahan masalah.

1.2 Mengapa Belajar R?

Berikut ini adalah beberapa alasan mengapa penting untuk belajar R:

1. **Pengolahan Data:** R dapat membantu Anda membersihkan, merubah format, dan mengolah data sebelum analisis lebih lanjut.
2. **Analisis Data:** R adalah alat yang kuat untuk menganalisis data, membuat visualisasi yang menarik, dan mengidentifikasi pola dalam dataset.
3. **Karir di Ilmu Data:** Penguasaan R menjadi salah satu keahlian yang sangat dihargai dalam industri ilmu data dan analisis data.
4. **Komunitas Besar:** Anda akan menjadi bagian dari komunitas besar yang mendukung dan berkontribusi dalam pengembangan R serta membagikan pengetahuan.

1.3 Download R & Rstudio:

1. **Unduh dan Instalasi:** Kunjungi situs resmi R (https://www.r-project.org/) untuk mengunduh installer sesuai dengan sistem operasi Anda.
2. **RStudio (Opsional tapi Disarankan):** RStudio adalah lingkungan pengembangan terintegrasi (IDE) yang mempermudah pengembangan dalam R. Anda dapat mengunduh RStudio (https://www.rstudio.com/) dan menggunakannya untuk menulis dan menjalankan kode R.

1.4 Tutorial Instal R & Studio

Berikut adalah panduan langkah demi langkah untuk menginstal R dan RStudio

1.4.1 Instalasi R

R adalah bahasa pemrograman inti yang digunakan oleh RStudio. Ikuti langkah-langkah di bawah ini untuk menginstal R:

Windows

1. Kunjungi situs resmi R di https://cran.r-project.org/mirrors.html.
2. Pilih cermin (mirror) terdekat untuk mengunduh installer R.

3. Unduh installer R untuk Windows dan jalankan file installer yang diunduh.
4. Ikuti panduan instalasi, pilih opsi default kecuali jika Anda tahu persis apa yang Anda lakukan.
5. Setelah instalasi selesai, R akan terinstal di komputer Anda.

MacOS

1. Kunjungi situs resmi R di https://cran.r-project.org/mirrors.html.
2. Pilih cermin (mirror) terdekat untuk mengunduh installer R.
3. Unduh installer R untuk macOS dan jalankan file installer yang diunduh.
4. Ikuti panduan instalasi, pilih opsi default kecuali jika Anda tahu persis apa yang Anda lakukan.
5. Setelah instalasi selesai, R akan terinstal di komputer Anda.

Linux

Di sistem Linux, Anda dapat menggunakan perintah terminal untuk menginstal R. Berikut adalah contoh untuk beberapa distribusi umum:

Ubuntu/Debian:

Buka Program `csharp` anda dan run koding dibawah ini!

```
Copy code
sudo apt-get update
sudo apt-get install r-base
```

CentOS/Fedora:

Buka Program `Command Prompt` anda dan run koding dibawah ini!

```
sudo yum install R
```

1.4.2 Instalasi RStudio:

RStudio adalah Integrated Development Environment (IDE) yang mempermudah pengembangan dalam R. Ikuti langkah-langkah di bawah ini untuk menginstal RStudio:

Windows, macOS, dan Linux:

1. Kunjungi situs resmi RStudio di https://www.rstudio.com/products/rstudio/download/.

2. Pilih “RStudio Desktop” yang sesuai dengan sistem operasi Anda.
3. Unduh installer RStudio dan jalankan file installer yang diunduh.
4. Ikuti panduan instalasi dan pilih opsi default kecuali jika Anda tahu persis apa yang Anda lakukan.
5. Setelah instalasi selesai, RStudio akan terinstal di komputer Anda.

1.5 Video Instalasi R & RStudio

1.5.1 Windows

1.5.2 MacOS

1.6 Interface R & RStudio:

Interface adalah tampilan aplikasi R dan Rstudio yang telah terpasang diperlihatkan pada Gambar 1.1 dan Gambar 1.2.

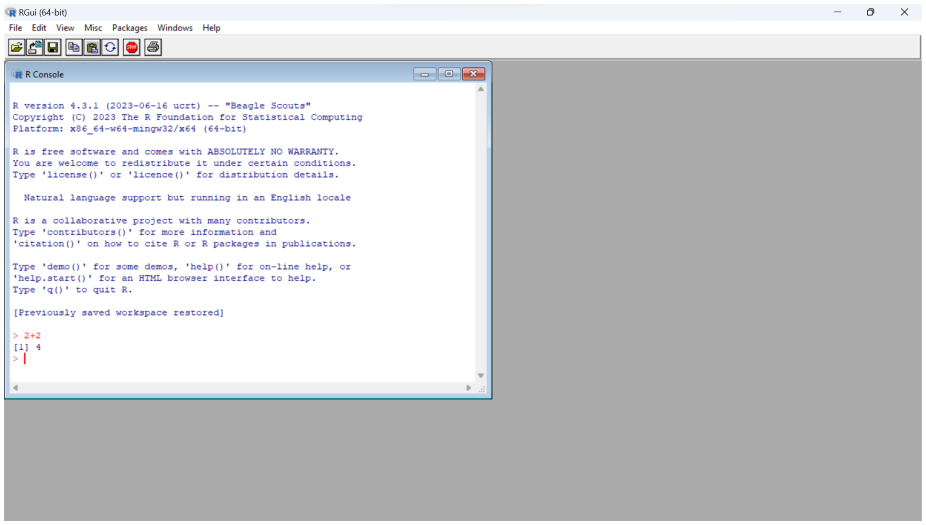

Figure 1.1: Jendela R.

Tampilan ini memiliki beberapa komponen utama, termasuk:

- **Script Panel:** Tempat Anda menulis kode R dalam skrip.
- **Console Panel:** Tempat hasil dari kode R ditampilkan, serta tempat Anda dapat menjalankan kode secara interaktif.
- **Environment Panel:** Menampilkan daftar variabel yang ada dalam sesi R Anda.

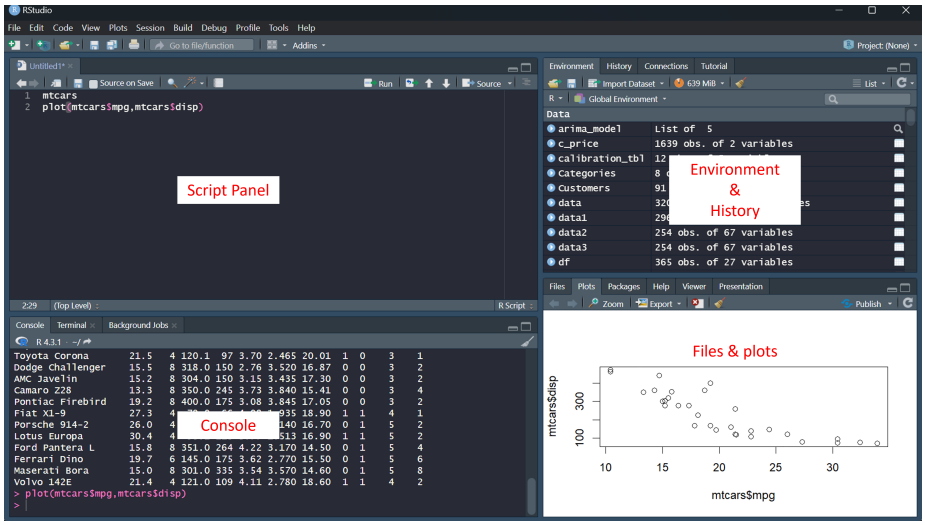

Figure 1.2: Jendela RStudio.

- **History Panel:** Menampilkan riwayat perintah yang telah dijalankan.
- **Files/Plots/Packages/Help Panel:** Panel tambahan yang membantu Anda mengelola file, visualisasi, pustaka, dan panduan bantuan.

Interface ini memudahkan pengguna untuk menulis, menjalankan, dan mengelola kode R serta menganalisis data dengan nyaman.

1.7 Sintaks Dasar R

Berikut ini beberapa kode sederhana yang bisa dipelajari untuk memulai memahami cara kerja Bahasa pemrograman R.

```
3+7
3-7
3^7
3/7
3*7
9^(1/3)
```

1.8 Bantuan (Help) R

Salah satu bagian terpenting dalam bekerja dengan bahasa R adalah mengetahui di mana mencari bantuan. R memiliki beberapa fasilitas in-line, selain berbagai

sumber daya bantuan di ekosistem R. Anda dapat menggunakan bantuan untuk fungsi tertentu.

```
help.start()          # menu di mana Anda dapat menavigasi bantuan lokal berbasis web
?help                 # menu di mana Anda dapat menavigasi bantuan lokal berbasis web
?class                # mendapatkan bantuan untuk fungsi `class`
help(class)           # mendapatkan bantuan untuk fungsi `class`
??class               # jika Anda tidak tahu nama fungsi yang Anda cari
help.search('class') # jika Anda tidak tahu nama fungsi yang Anda cari
```

1.9 Shortcut Penggunaan Rstudio

Beberapa petunjuk bermanfaat untuk Rstudio (IDE) meliputi:

Kata Kunci	Perintah	Detail
Ctrl + Return (Enter)	untuk menjalankan baris dari editor	~
Ctrl + Shift + #	untuk fokus pada tab bantuan	kontradiktif
Alt + Shift + k	untuk jalur pintas keyboard RStudio	~
Ctrl + r	untuk menelusuri sejarah perintah	~
Alt + Shift + j	untuk menavigasi antar bagian kode	~
Ctrl + 1	untuk melompat ke editor	tab untuk penyelesaian otomatis
Ctrl + 2	untuk melompat ke konsol	tab untuk penyelesaian otomatis
Ctrl + 8	untuk melompat ke environment list	tab untuk pelengkapan otomatis
Alt + l	Collapse chunk	Code Folding
Alt + Shift + l	Unfold chunk	Code Folding
Alt + o	Collapse all	Code Folding
Alt + Shift + o	Unfold all	Code Folding
Alt + “-”	untuk operator penugasan <-	~
Alt + Shift + c	kode komentar/tanda komentar dalam file	.R kontradiktif

1.10 Praktikum

Buatlah tutorial Instalasi R dan R Studio dalam M.word! Lengkapi setiap prosesnya dengan gambar dan penjelasan.

Chapter 2

Operasi Dasar R

Pemrograman R merujuk pada proses menulis kode dan mengembangkan program menggunakan bahasa pemrograman R. R adalah bahasa pemrograman yang fokus pada analisis statistik, manipulasi data, dan visualisasi. Pada bab ini akan dibahas beberapa unsur utama dalam pemrograman menggunakan bahasa pemrograman R.

2.1 Variabel

Variabel dalam bahasa pemrograman R digunakan untuk menyimpan dan mengelola data. Variabel memungkinkan Anda untuk menampung nilai-nilai berbagai jenis, seperti angka, karakter (teks), atau nilai logika (benar/salah). Berikut ini adalah cara untuk mendefinisikan dan menggunakan variabel dalam R:

2.1.1 Mendefinisikan Variabel

Untuk membuat variabel, Anda cukup menggunakan tanda <- atau = untuk memberikan nilai pada variabel.

```
x <- 10          # Mendefinisikan variabel x
y = 12           # Mendefinisikan variabel y
```

2.1.2 Aturan Nama Variabel

- Nama variabel harus dimulai dengan huruf atau tanda .
- Nama variabel bisa terdiri dari huruf, angka, dan tanda _.

- Karakter khusus seperti +, -, *, /, ^ dll. tidak diperbolehkan dalam nama variabel.
- Nama variabel bersifat case-sensitive, artinya `x` dan `X` dianggap berbeda.

2.2 Operator

Operator adalah simbol yang mengarahkan compiler untuk melakukan berbagai macam operasi terhadap beberapa penugasan. Operator mensimulasikan berbagai operasi matematis, logika, dan keputusan yang dilakukan pada sekumpulan Bilangan Kompleks, Integer, dan Numerik sebagai penugasan masukan (input). R mendukung sebagian besar empat jenis operator biner antara satu set penugasan. Dalam ini, kita akan melihat berbagai jenis operator yang tersedia di R penggunaannya.

2.2.1 Aritmatika

Penggunaan operator aritmatika dalam program R adalah untuk mensimulasikan berbagai operasi matematika, seperti penambahan, pengurangan, perkalian, pembagian, dan modulo. Operator aritmatika yang dilakukan bisa saja berupa nilai skalar, bilangan kompleks, atau vektor.

Operator	**R**
Penjumlahan	+
Pengurangan	-
Perkalian	*
Divisi/Pembagian	/
Pemangkatan	^
Modulo	%%

Perhatikan cuplikan R berikut:

```
x <- c(2,3,5)     # memuat vektor x
y <- c(2,4,6)     # memuat vektor y
x+y               # hasil penjumahan vektor x dan y
print (x+y)       # hasil penjumahan vektor x dan y
print (x-y)       # hasil pengurangan vektor x dan y
print (x*y)       # hasil perkalian vektor x dan y
print (x/y)       # hasil pembagian vektor x dan y
print (x^y)       # hasil pemangkatan vektor x dan y
print (x%%y)      # hasil modulo vektor x dan y
```

Adakalanya anda perlu menampilkan keterangan/komentar yang juga melekat pada hasil perhitungan R itu sendiri. Maka anda dapat melakukannya dengan cara berikut:

```
cat("Penjumahan vektor x dan y :", x + y, "\n")
cat("Pengurangan vektor x dan y :", x - y, "\n")
cat("Perkalian vektor x dan y :", x * y, "\n")
cat("Pembagian vektor x dan y :", x / y, "\n")
cat ("Pemangkatan vektor x dan y :", x ^ y)
cat("Modulo vektor x dan y :", x %% y, "\n")
```

Catatan: Penjelasan lebih lekap mengenai modulo dapat lihat pada link ini

2.2.2 Relasional

Operator relasional melakukan operasi perbandingan antara elemen yang bersesuaian pada setiap operan. Mengembalikan nilai Boolean TRUE jika operan pertama memenuhi relasi dibandingkan dengan operan kedua. Nilai TRUE selalu dianggap lebih besar dari FALSE.

Operator	R	Keterangan
Kurang dari	<	Mengembalikan TRUE jika elemen yang bersesuaian pada operan pertama lebih kecil dari operan kedua. Selain itu akan mengembalikan FALSE
Kurang dari sama dengan	<=	Mengembalikan TRUE jika elemen yang bersesuaian pada operan pertama kurang dari atau sama dengan elemen operan kedua. Selain itu akan mengembalikan FALSE
Lebih besar dari	>	Mengembalikan TRUE jika elemen yang bersesuaian pada operan pertama lebih besar dari operan kedua. Selain itu akan mengembalikan FALSE

Lebih besar dari sama dengan	>=	Mengembalikan BENAR jika elemen yang bersesuaian pada operan pertama lebih besar atau sama dengan dari operan kedua. Selain itu akan mengembalikan FALSE
Sama Dengan	==	Mengembalikan BENAR jika dan hanya jika kedua sisi bernilai sama
Tidak Sama dengan	!=	Mengembalikan BENAR jika elemen yang bersesuaian pada operan pertama tidak sama dengan dari operan kedua

```
x <- c(2,3,5)     # memuat vektor x
y <- c(2,4,6)     # memuat vektor y
cat("Vektor x  kurang dari Vektor y:", x < y, "\n")
cat("Vector x kurang dari sama dengan Vector y:", x <= y, "\n")
cat("Vector x lebih besar dari Vector y :", x > y, "\n")
cat("Vector x lebih besar dari sama dengan Vector y :", x >= y, "\n")
cat("Vector x sama dengan Vector y:", x == y,"\n")
cat("Vector x tidak sama dengan Vector y:", x != y)
```

2.2.3 Logika

Operator logis mensimulasikan operasi keputusan, berdasarkan operator yang ditentukan antara operan, yang kemudian dievaluasi ke nilai Boolean Benar atau Salah. Nilai bilangan bulat bukan nol dianggap sebagai nilai BENAR, baik itu bilangan kompleks atau bilangan real.

Operator	**R**	**Keterangan**
NOT	!	Operasi negasi/kebalikan pada status elemen operan
AND	&	Mengembalikan TRUE jika kedua operan bernilai Benar
OR	|	Mengembalikan TRUE jika salah satu operan adalah Benar

XOR	^	Mengembalikan TRUE jika salah satu dari kedua elemen pertama operan bernilai Benar

```
x <- c(0,TRUE,FALSE)
y <- c(TRUE,0.1,4+3i)

# Melakukan operasi logika pada Operan
cat("Logika Negasi (~) untuk vektor x:", !x, "\n")
cat("Logika Negasi (~) untuk vektor y :", !y, "\n")
cat ("Logika Konjungsi (Dan) :", x & y, "\n")
cat ("Logika Disjungsi (Atau) :", x | y, "\n")
```

2.2.4 Operator Lain-lain

Berikut ini juga ada beberapa operator yang kemungkinan besar juga akan anda perlukan pada saat akan menggunakan R.

```
x <- c(2,3,5)                    # memuat vektor x
y <- c(2,4,6)                    # memuat vektor y
sqrt(x*y)                        # Bentuk akar
log(x)                           # logaritma
exp(y)                           # eksponen
(x/y) + y                        # Tanda kurung
```

Catatan: Sifat Komutatif Asosiatif dan Distributif juga berlaku dalam program R.

2.3 Tipe Data

Dalam pemrograman seperti R dan Python, tipe data merupakan konsep penting. Keduanya dapat menggunakan variabel untuk menyimpan tipe yang berbeda-beda, berikut adalah tipe data paling mendasar yang harus diketahui:

Tipe Data	**R**	**Penjelasan**
Double/Float	5.6	Bilangan yang mempunyai koma
Integer	5	Bilangan bulat 1,2,…,n

Bolean/Logical	TRUE/FALSE	Benar bernilai 1 dan Salah bernilai 0
String/Character	'Dsciencelabs'	karakter/kalimat bisa berupa huruf angka, dll (diapit tanda " atau ')
Complex	1 + 5i	Pasangan angka real dan imajiner

Berikut ini adalah koding R yang dapat digunakan untuk menetapkan kelima tipe data diatas:

```
d1 = 5.6                          # Tetapkan nilai desimal
d2 = as.integer(5)                # tetapkan nilai integer
d2 = 5L                           # cara lain untuk memuat nilai integer di R
d3 = c(TRUE,FALSE)                # Bolean/Logical
d3 = as.logical(c(0,1))           # cara lain untuk memuat Bolean/Logical
d4 = c("a",'b','123')             # String/Character
d5 = 1 + 5i                       # Complex
```

Untuk memeriksa tipe data dalam R:

```
class(d1)                         # cetak nama kelas variabel
typeof(d1)                        # cetak tipe variabel x
```

2.4 Statistika Dasar

```
data <- c(85, 90, 78, 92, 88)  # Data
sum(data)                      # Jumlahan data
length(data)                   # Banyaknya data
mean (data)                      # Menghitung rata-rata
var (data)                     # Menghitung variansi
sd (data)                        # Simpangan baku
min(data)                      # Minimum
max(data)                      # Maksimum
```

2.5 Contoh Pemrograman Dasar

Berikut ini dilampirkan contoh kasus dasar pemrograman R.

2.5.1 Menghitung Rata-rata

```
data <- c(85, 90, 78, 92, 88)
jumlah_data <- length(data)
total <- sum(data)
rata_rata <- total / jumlah_data
print(paste("Rata-rata:", rata_rata))
```

```
## [1] "Rata-rata: 86.6"
```

2.5.2 Membandingkan Angka

```
a <- 10
b <- 20

if (a < b) {
  print("a lebih kecil dari b")
} else if (a > b) {
  print("a lebih besar dari b")
} else {
  print("a sama dengan b")
}
```

```
## [1] "a lebih kecil dari b"
```

2.5.3 Membandingkan Karakter

```
kata1 <- "Apel"
kata2 <- "apel"

if (kata1 == kata2) {
  print("kata1 sama dengan kata2")
} else {
  print("kata1 berbeda dari kata2")
}
```

```
## [1] "kata1 berbeda dari kata2"
```

2.5.4 Mengecek Kondisi Gabungan

```
umur <- 25
pendapatan <- 5000

if (umur > 18 && pendapatan > 4000) {
  print("Anda memenuhi syarat")
} else {
  print("Anda tidak memenuhi syarat")
}
```

```
## [1] "Anda memenuhi syarat"
```

2.5.5 Penggunaan Operator Logika

```
x <- 5
y <- 10

if (x > 0 || y > 0) {
  print("Salah satu variabel positif")
} else {
  print("Kedua variabel non-positif")
}
```

```
## [1] "Salah satu variabel positif"
```

2.5.6 Pemeriksaan Kondisi dengan ifelse()

```
nilai <- 75
keterangan <- ifelse(nilai >= 70, "Lulus", "Tidak lulus")
print(paste("Nilai:", nilai, "--> Status:", keterangan))
```

```
## [1] "Nilai: 75 --> Status: Lulus"
```

2.6 Latihan

Berikut adalah beberapa contoh soal latihan dasar pemrograman dalam bahasa pemrograman R:

1. **Menghitung Luas Lingkaran:** Buatlah sebuah program R yang menerima input berupa jari-jari lingkaran dan menghitung serta mencetak luas lingkaran.
2. **Konversi Suhu:** Buatlah sebuah program R yang dapat mengonversi suhu dari Celsius ke Fahrenheit. Program harus menerima input suhu dalam Celsius dan menghasilkan output suhu dalam Fahrenheit.
3. **Menghitung Faktorial:** Buatlah sebuah program R yang menghitung faktorial dari sebuah bilangan bulat positif. Program harus menerima input bilangan bulat positif dan menghasilkan output faktorialnya.
4. **Mencari Bilangan Prima:** Buatlah sebuah program R yang menerima input sebuah bilangan bulat dan menghasilkan output apakah bilangan tersebut merupakan bilangan prima atau bukan.
5. **Menghitung Pangkat:** Buatlah sebuah program R yang menerima input bilangan dasar dan eksponen, kemudian menghitung hasil dari bilangan dasar dipangkatkan dengan eksponen tersebut.
6. **Menghitung Total Nilai:** Buatlah sebuah program R yang menerima input sejumlah nilai mata kuliah dan menghitung total nilai serta rata-ratanya. Program harus menerima input nilai-nilai mata kuliah dan menghasilkan total nilai serta rata-ratanya.
7. **Menentukan Ganjil/Genap:** Buatlah sebuah program R yang menerima input bilangan bulat dan mencetak apakah bilangan tersebut ganjil atau genap.
8. **Menghitung Keliling dan Luas Persegi:** Buatlah sebuah program R yang menerima input panjang sisi persegi dan menghitung serta mencetak keliling dan luasnya.

Chapter 3

Struktur Data

Struktur data dalam R adalah cara di mana Anda dapat mengatur dan menyimpan data dalam bentuk yang terstruktur agar mudah diakses, dikelola, dan dimanipulasi. Struktur data memungkinkan Anda untuk mengelompokkan nilai-nilai data ke dalam objek yang sesuai dengan jenis dan sifat data yang Anda miliki. R memiliki beberapa jenis struktur data yang dapat digunakan untuk berbagai tujuan.

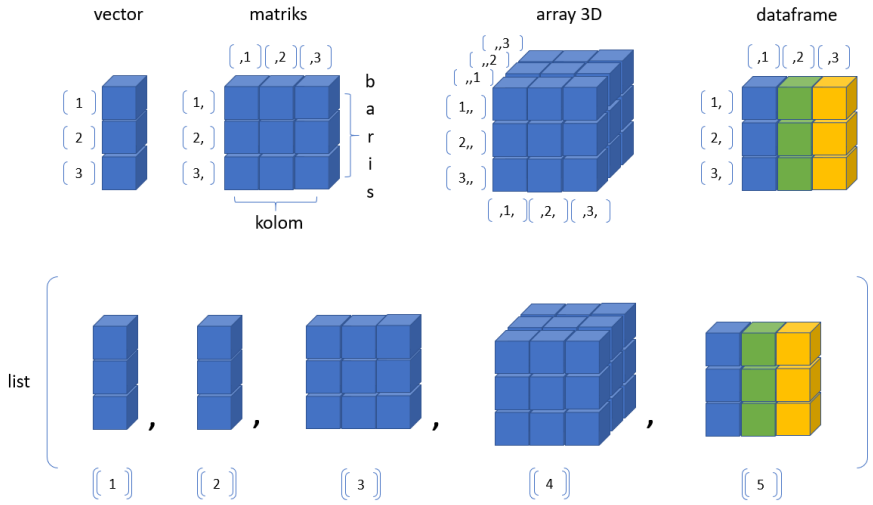

Figure 3.1: Struktur Data dalam R

3.1 Vektor

Elemen paling dasar dalam R adalah vektor, yang berisikan kumpulan elemen data dengan tipe yang sama. Terdapat dua jenis vector, yaitu vector numerik dan vector karakter. Misalnya:

```
vektor <- seq(from=10, to=21, by=1)       # fungsi `seq()` dengan "by"
vektor<- seq(from=10, to=21, len=12)      # fungsi `seq()` dengan "len"
vektor <- 10:21                           # tetapkan data dalam vektor
vektor <- vektor+2                        # Operasi berdasarkan elemen
vektor <- vektor*2                        # Tambahkan 2 untuk setiap elemen
vektor <- vektor^2                        # Pangkat 2 untuk setiap elemen
vektor <- sqrt(vektor)                    # Akar kuadrat untuk setiap elemen
vektor <- log(vektor)                     # Logaritma untuk setiap elemen
vektor<- c(0.5, 0.6)                      # Numerik
vektor <- c(TRUE, FALSE)                  # Logis
vektor <- c(T, F)                         # Logis
vektor <- c("a", "b", "c")                # Karakter
vektor <- 9:29                            # Integer
vektor <- c(1+0i, 2+4i)                   # Kompleks
vektor <- vector("numeric", length = 10) # untuk inisialisasi vektor.
```

Catatan: Menurut dokumentasi R untuk `typeof()` dan `class()`, pernyataan tentang "perbedaan utama/main difference" adalah tidak benar. Kelas adalah atribut dari objek yang dapat ditetapkan terlepas dari mode penyimpanan internalnya, sedangkan `typeof()` menentukan tipe (R internal) atau mode penyimpanan dari objek apa pun. Salah satu menggambarkan karakteristik logis sedangkan yang lain adalah karakteristik fisik dari suatu objek.

```
class(vektor)                             # Periksa kelas vektor
as.numeric(vektor)                        # Menetapkan vektor sebagai numerik
as.logical(vektor)                        # Menetapkan vektor sebagai logis
as.character(vektor)                      # Menetapkan vektor sebagai karakter
as.numeric(c(FALSE,TRUE,TRUE,FALSE))      # Menetapkan vektor logis sebagai angka
```

Terkadang, R tidak dapat menemukan cara untuk memaksa suatu objek dan ini dapat menghasilkan NA.

```
vektor <- c("a", "b", "c","1")              # menetapkan nilai vektor
as.numeric(vektor)                        # menetapkan vektor sebagai numerik
```

```
## Warning: NAs introduced by coercion
```

```
as.logical(vektor)                    # menetapkan vektor sebagai logis
as.complex(vektor)                    # menetapkan vektor sebagai karakter
```

```
## Warning: NAs introduced by coercion
```

Catatan: Saat paksaan tidak masuk akal terjadi, Anda biasanya akan mendapat peringatan dari R.

Kita sudah melihat bahwa elemen dasar dari objek R adalah vektor. Vektor dapat ditetapkan dengan berbagai jenis berikut:

- **character:** di mana setiap elemen adalah string, mis., urutan simbol alfanumerik.
- **numeric:** di mana setiap elemen adalah bilangan real dalam format floating point presisi ganda.
- **integer:** di mana setiap elemen adalah integer.
- **logis:** di mana setiap elemen adalah TRUE, FALSE, atau NA3
- **complex:** di mana setiap elemen adalah bilangan kompleks.

3.2 Matriks

Matriks adalah vektor dengan atribut dimensi. Matriks dibuat berdasarkan kolom, sehingga entri dapat dianggap dimulai dari sudut "kiri atas" dan mengalir di kolom.

```
matriks <- matrix(c(1, 2, 3, 4, 5, 6), nrow = 2, ncol = 3)
matriks
```

```
##      [,1] [,2] [,3]
## [1,]    1    3    5
## [2,]    2    4    6
```

Matriks juga dapat dibuat langsung dari vektor dengan menambahkan atribut dimensi.

```
matriks <- 1:6                        # Membuat vektor
dim(matriks) <- c(2, 3)               # rubah vektor sebagai matriks sebesar 2x3
matriks                               # Mencetak hasilnya
```

Matriks dapat dibuat dengan pengikatan kolom atau pengikatan baris dengan fungsi `cbind()` dan `rbind()`.

```r
x <- 1:3                                    # Membuat vektor `x`
y <- 10:12                                  # Membuat vektor `y`
cbind(x, y)                                 # Menggabungkan vektor `x` dan` y` dengan kolom
rbind(x, y)                                 # Menggabungkan vektor `x` dan` y` dengan baris
```

3.3 Array

Array mirip dengan matrix, tetapi dapat memiliki lebih dari dua dimensi. Masing-masing dimensi dalam array memiliki ukuran tertentu.

```r
array_data <- array(c(1, 2, 3, 4, 5, 6), dim = c(2, 3, 1))
```

3.4 Faktor

Faktor-faktor digunakan untuk mewakili data kategorikal dan dapat menjadi tidak teratur atau teratur. Orang dapat menganggap faktor sebagai vektor integer di mana setiap integer memiliki label. Menggunakan faktor dengan label lebih baik daripada menggunakan bilangan bulat karena faktor menggambarkan diri sendiri. Memiliki variabel yang memiliki nilai "Laki-laki" dan "Perempuan" lebih baik daripada variabel yang memiliki nilai 1 dan 2. Objek-objek dapat dibuat dengan fungsi `faktor()`.

```r
x <- factor(c("yes","no","yes","no"))  # Membuat objek faktor
x                                      # Cetak hasilnya
table(x)                               # Tabel dari `x`
unclass(x)                             # Melihat representasi faktor yang mendasarinya
attr(x,"levels")                       # Melihat representasi faktor yang mendasarinya
```

3.5 Data Frame

Kerangka data (data frame) adalah tabel atau struktur mirip array dua dimensi di mana setiap kolom berisi nilai satu variabel dan setiap baris berisi satu set nilai dari setiap kolom.

Berikut ini adalah karakteristik data frame.

- Nama kolom tidak boleh kosong;
- Nama baris harus unik;
- Data yang disimpan dalam data frame bisa dari numerik, faktor atau tipe karakter;

- Setiap kolom harus berisi jumlah item data yang sama.

```
# Buat data frame pertama.
df1 <- data.frame(id = c (1:5),
                  name = c("Julian","Vanessa","Jeffry","Angel","Nikki"),
                 salary = c(623.3,515.2,611.0,729.0,843.25),
              start_date = as.Date(c("2022-01-01", "2022-09-23", "2022-11-15",
                    dept = c("DS","DS","BA","DA","DS"), stringsAsFactors = F)
df1
```

```
# Buat data frame kedua.
df2 <-data.frame(id = c (6:10),
                 name = c("Ardifo","Irene","Kefas","Sherly","Bakti"),
                salary = c(578.0,722.5,632.8,632.8,NA),
            start_date = as.Date(c("2022-05-21","2022-07-30","2022-06-17",
                                   "2022-07-30","2018-09-03")),
                 dept = c("Actuaries","Actuaries","CA","DE","Lecturer"),stringsAsFactors = F)
df2
```

```
df3 <- rbind(df1,df2)                    # Gabungkan dua frame data
print(df3)                               # Cetak hasilnya `df3`
head(df3)                                # Cetak enam baris pertama
head(df3,6)                              # Cetak enam baris pertama
#View(df3)                               # Menggunakan RStudio seperti penampil Excel
class(df3)                               # objeknya bertipe data.frame
str(df3)                                 # Dapatkan struktur data frame
dim(df3)                                 # Periksa dimensi data
```

Data frame biasanya dibuat dengan membaca dalam dataset menggunakan `read.table()` atau `read.csv ()`. Namun, data frame juga dapat dibuat secara eksplisit dengan fungsi `data.frame()` atau mereka dapat dipaksakan dari jenis objek lain seperti list.

3.6 Lists

List dalam R adalah struktur data yang mengizinkan Anda untuk menyimpan berbagai jenis objek, termasuk vektor, matriks, array, dataframe, dan objek list lainnya, dalam satu objek tunggal. Ini memungkinkan Anda untuk membuat struktur data yang kompleks dan fleksibel dengan menggabungkan objek-objek yang berbeda ke dalam satu wadah. List sering digunakan ketika Anda perlu mengorganisir dan mengelompokkan objek-objek yang terkait.

Berikut adalah contoh penggunaan dan pembuatan list dalam R:

```r
# Membuat vektor dan matriks
vektor <- c(1.5, 2.7, 3.2, 4.0)
matriks <- matrix(c(1, 2, 3, 4, 5, 6), nrow = 2, ncol = 3)
```

```r
# Membuat dataframe
data_frame <- data.frame(name = c("Alice", "Bakti", "Charlie"),
                         age = c(25, 30, 28),
                         score = c(95, 88, 76))
```

```r
faktor <- "List, Sudah Jadi"
```

```r
# Membuat list
my_list <- list(vektor, matriks, data_frame, faktor)
```

```r
# Menampilkan list
print(my_list)
```

Anda juga dapat memberi nama pada setiap elemen dalam list untuk membuat list yang lebih mudah dibaca:

```r
nama_list <- list(elemen1 = vektor,
                  elemen2 = matriks,
                  elemen3 = data_frame,
                  elemen4 = faktor)

# Menampilkan elemen dalam list berdasarkan nama
print(nama_list$elemen1)
```

```
## [1] 1.5 2.7 3.2 4.0
```

```r
print(nama_list$elemen2)
```

```
##      [,1] [,2] [,3]
## [1,]    1    3    5
## [2,]    2    4    6
```

```r
print(nama_list$elemen3)
```

```
##      name age score
## 1   Alice  25    95
## 2   Bakti  30    88
## 3 Charlie  28    76
```

```
print(nama_list$elemen4)
```

```
## [1] "List, Sudah Jadi"
```

Anda dapat mengakses elemen-elemen dalam list menggunakan indeks atau nama. Misalnya:

```
# Mengakses elemen pertama dalam list menggunakan indeks
elemen1 <- my_list[[1]]
elemen2 <- my_list[[2]]
elemen3 <- my_list[[2]]
elemen4 <- my_list[[2]]

# Menampilkan hasil
print(elemen1)
print(elemen2)
print(elemen3)
print(elemen4)
```

List memungkinkan Anda mengorganisir, mengelompokkan, dan mengakses objek-objek yang beragam dalam struktur data tunggal, sehingga sangat berguna dalam analisis data yang kompleks dan beragam.

3.7 Rekayasa Data Frame

3.7.1 Tanpa Packages

Sebagai seorang Data Scientist, ketika mencoba menyimulasikan proses analisis data, pemodelan, bahkan prediksi. Anda harus mampu secara intuitif membangun dataframe untuk memperkirakan kumpulan data sampel. Terutama, ketika Anda tidak memiliki kumpulan data sampel sama sekali. Oleh karena itu, pada bagian ini, kita akan belajar sedikit mengenai cara menghasilkan dataframe. Harap perhatikan baik-baik contoh berikut:

```
# Misalkan Anda ingin membangun kumpulan data karyawan di sebuah perusahaan

No<-(1:52)                                    # Menghasilkan bilangan 1-52
Name<-c(LETTERS,letters)                      # 26 LETTERS dan 26 letters
Gender<-sample(rep(c("Male","Female"),times=26)) # 26 Laki-laki dan 26 perempuan

# Menghasilkan tanggal lahir
year_in_3<-seq(as.Date("2000/01/01"), by="year", length.out=4)
```

```
Birthday <- rep(year_in_3, times=13)

# Menghasilkan kategori universitas
univ1<-rep("National",times=26)                  # 26 universitas negeri
univ2<-rep("Private",times=16)                   # 16 universitas swasta
univ3<-rep("Overseas",times=10)                  # 10 universitas luar negeri
Universities<-sample(c(univ1,univ2,univ3))       # Menggabungkan data (vetor)

gpa<-runif(52,min=3.00,max=4.00)                 # Menghasilkan 52 bilangan acak (min=
GPA<-round(gpa,digits=2)                         # Mengatur digit bilangan acak Anda
Salary<-sample(600:1200,52,replace=T)            # Menghasilakn sampel antara 600-1200
Employees<-data.frame(No,
                      Name,
                      Birthday,
                      Gender,
                      Universities,
                      GPA,
                      Salary)
Employees
```

```
##    No Name   Birthday Gender Universities  GPA Salary
## 1   1    A 2000-01-01 Female     National 3.43    746
## 2   2    B 2001-01-01   Male     National 3.49   1043
## 3   3    C 2002-01-01 Female      Private 3.59    714
## 4   4    D 2003-01-01 Female     National 3.54   1115
## 5   5    E 2000-01-01 Female     National 3.75    629
## 6   6    F 2001-01-01   Male      Private 3.42    934
## 7   7    G 2002-01-01   Male     National 3.18    853
## 8   8    H 2003-01-01 Female     Overseas 3.13    813
## 9   9    I 2000-01-01   Male      Private 3.96   1060
## 10 10    J 2001-01-01 Female     National 3.86    879
## 11 11    K 2002-01-01 Female      Private 3.38    933
## 12 12    L 2003-01-01 Female     Overseas 3.14    824
## 13 13    M 2000-01-01 Female     National 3.65    745
## 14 14    N 2001-01-01 Female     Overseas 3.14   1181
## 15 15    O 2002-01-01   Male     National 3.22    980
## 16 16    P 2003-01-01   Male     Overseas 3.04   1040
## 17 17    Q 2000-01-01   Male      Private 3.13    963
## 18 18    R 2001-01-01 Female      Private 3.42    662
## 19 19    S 2002-01-01   Male      Private 3.94   1065
## 20 20    T 2003-01-01 Female     National 3.74   1092
## 21 21    U 2000-01-01   Male     National 3.13    989
## 22 22    V 2001-01-01   Male      Private 3.34   1101
## 23 23    W 2002-01-01 Female     National 3.90    776
## 24 24    X 2003-01-01 Female     Overseas 3.94    895
```

```
## 25 25    Y 2000-01-01   Male     National 3.31    956
## 26 26    Z 2001-01-01   Male     National 3.71    615
## 27 27    a 2002-01-01 Female     Overseas 3.14    819
## 28 28    b 2003-01-01 Female     Overseas 3.78   1177
## 29 29    c 2000-01-01   Male     National 3.03    997
## 30 30    d 2001-01-01   Male     National 3.37    663
## 31 31    e 2002-01-01   Male     Overseas 3.89    702
## 32 32    f 2003-01-01 Female     National 3.87    808
## 33 33    g 2000-01-01 Female      Private 3.75   1034
## 34 34    h 2001-01-01   Male     National 3.31    678
## 35 35    i 2002-01-01 Female      Private 3.03    823
## 36 36    j 2003-01-01 Female      Private 3.39    810
## 37 37    k 2000-01-01   Male     National 3.07    983
## 38 38    l 2001-01-01 Female     Overseas 3.57    743
## 39 39    m 2002-01-01   Male      Private 3.25   1107
## 40 40    n 2003-01-01 Female      Private 3.36    731
## 41 41    o 2000-01-01 Female     National 3.01   1054
## 42 42    p 2001-01-01   Male     Overseas 3.82    678
## 43 43    q 2002-01-01   Male     National 3.93    619
## 44 44    r 2003-01-01 Female     National 3.77    974
## 45 45    s 2000-01-01   Male     National 3.93    859
## 46 46    t 2001-01-01   Male      Private 3.81    924
## 47 47    u 2002-01-01 Female     National 3.97    664
## 48 48    v 2003-01-01   Male     National 3.12   1087
## 49 49    w 2000-01-01 Female      Private 3.69   1103
## 50 50    x 2001-01-01   Male     National 3.76    720
## 51 51    y 2002-01-01   Male     National 3.31    734
## 52 52    z 2003-01-01   Male      Private 3.21    895
```

3.7.2 Mengunakan Packages

Dalam contoh kedua ini, digunakan pustaka *(Packages)* `faker` untuk menghasilkan data palsu seperti nama, alamat, dan lain-lain. Pastikan Anda telah menginstal pustaka tersebut menggunakan perintah `install.packages("fakir")` jika belum terinstal, mengikuti langkah berikut.

```
install.packages("remotes")
remotes::install_github("ThinkR-open/fakir")
```

Selanjutnya, anda dapat membuat data frame palsu seperti diperlihatkan berikut:

```
library(fakir)
fake_ticket_client(vol = 10)
```

```
## # A tibble: 10 x 25
##    ref        num_client first last  job     age region
##    <chr>      <chr>      <chr> <chr> <chr> <dbl> <chr>
##  1 DOSS-AMQN~ 79         Jovan O'Ke~ Gene~    22 Auver~
##  2 DOSS-NCKJ~ 69         Miss  Lean~ Emer~    68 Bourg~
##  3 DOSS-GPBE~ 120        Odell Stok~ Engi~    24 Prove~
##  4 DOSS-GRLN~ 31         Loren Lars~ <NA>     NA Auver~
##  5 DOSS-LEPJ~ 59         Mayb~ Maye~ Furt~    18 Langu~
##  6 DOSS-DUCL~ 118        Jama~ Ober~ Engi~    18 Île-d~
##  7 DOSS-OCED~ 77         Lee   Scha~ Admi~    NA Limou~
##  8 DOSS-KXSJ~ 65         Deme~ Auer  Cont~    21 <NA>
##  9 DOSS-UITD~ 141        Wilf~ Harv~ Educ~    53 Rhône~
## 10 DOSS-SHKL~ 182        Addy~ Nien~ Earl~    65 Alsace
## # i 18 more variables: id_dpt <chr>,
## #   departement <chr>, cb_provider <chr>, name <chr>,
## #   entry_date <dttm>, fidelity_points <dbl>,
## #   priority_encoded <dbl>, priority <fct>,
## #   timestamp <date>, year <dbl>, month <dbl>,
## #   day <int>, supported <chr>,
## #   supported_encoded <int>, type <chr>, ...
```

Catatan: Pustaka `fakir` Menyimpan beberapa dataset didalamnya, antara lain:

```
fake_products(10)                                    # Rekayasa Data Produk
fake_visits(from = "2017-01-01", to = "2017-01-31") # Pengunjung Website
fake_sondage_answers(n = 10)                         # Kuisioner transfortasi
```

3.8 Latihan

1. Buatlah Rekayasa dataframe Mahasiswa dengan empat kolom: “Nama”, “Usia”, “Kota”, dan “Nilai”. Sebanyak 100 baris, dengan syarat tidak boleh ada nama yang sama.

2. Buatlah Rekayasa dataframe Karyawan dengan tujuh kolom: “No”, “Name”, “Birthday”, “Gender”, “Universities”, “GPA”, “Salary”. Sebanyak 100 baris, dengan syarat tidak boleh ada nama yang sama.

3. Buatlah Rekayasa dataframe pengunjung Website, sebanyak 200 baris.

Catatan: Kumpulkan hasil latihan anda, tidak boleh sama dengan teman mahasiswa lainnya.

Chapter 4

Ekstraksi Data

Ekstraksi data mengacu pada proses mengambil sebagian atau elemen tertentu dari struktur data seperti vektor, matriks, dataframe, atau list.

4.1 Impor Dataset

Berikut ini digunakan dataset yang sudah ada didalam R.

```
library(stima)                          # Pustaka Dataset
View(employee)                          # Data employee dari Pustaka stima
typeof(employee)                        # Memeriksa jenis data
class(employee)                         # Memeriksa kelas data
```

4.2 Ekstraksi Baris

Ekstraksi baris berdasarkan indeks dapat dilakukan dengan operator `[]`.

```
employee[1,5]                           # ekstrak elemen di baris 1 dan kolom 5
employee$gender                         # Mengekstrak kolom terntentu (`Gender`)
employee[,c('jobcat','gender')]         # Mengekstrak kolom tertentu (jobcat, gender`)
employee[1:5,]                          # Mengekstrak lima baris pertama dari employee
employee[,1:5]                          # Mengekstrak lima kolom pertama dari employee
employee$edu >12                        # Subset kolom dengan kondisi
employee[employee$edu>12 & employee$salary>40200,] # Subset Kondisi Kombinasi indeks
```

4.3 Ekstraksi Subset

Ekstraksi subset mengacu pada proses mengambil sebagian data dari suatu struktur data berdasarkan kriteria tertentu. Ini memungkinkan Anda untuk fokus pada subset data yang relevan untuk analisis atau tugas tertentu tanpa harus memanipulasi atau mengakses keseluruhan data. Subset dapat diambil berdasarkan kondisi tertentu, indeks, atau kombinasi keduanya.

Dalam R, Anda dapat melakukan ekstraksi subset menggunakan berbagai metode tergantung pada jenis struktur data yang Anda gunakan. Beberapa teknik umum untuk ekstraksi subset adalah:

```
subset(employee, select='jobcat')    # subset kolom tertentu
subset(employee, select=6)           # subset kolom tertentu
subset(employee, select= c(6,7))     # subset kolom pertama dan kedua
subset(employee, select= c(2:5))     # subset kolom tertentu
```

4.4 Ekstraksi Dengan Variabel Baru

Menambah variabel baru ke dalam dataframe dapat dilakukan dengan menetapkan vektor baru. Kekuatan objek dataframe adalah menerima hampir semua jenis vektor, mis. integer, numerik, logika, faktor, dan karakter.

```
# Tambahkan variabel baru 'Tax' ke dataframe Employess
employee$Tax <- employee$salary*0.03

# Tambahkan variabel baru 'Take_Home_Pay' ke dataframe Employess
employee$Take_Home_Pay <- employee$salary-employee$Tax

 # Tambahkan variabel baru 'Category_Salary1' ke dataframe Employess
employee$Category_Salary1 <- employee$salary > 92000

# Tambahkan variabel baru 'Category_Salary2' ke dataframe Employess
employee$Category_Salary2<-ifelse(employee$salary > 92000,
                                  "High",
                                  "Low")
employee$Category_Salary2<-factor(employee$edu >12 &
                                  employee$salary>92000,
                                  labels = c("level1",
                                             "level2"))
```

4.5 Ekstraksi Summary Statistik

```
min(employee$salary)                    # Temukan nilai minimum dari `Salary`
max(employee$salary)                    # Temukan nilai mksimum dari `Salary`
mean(employee$salary)                   # Temukan nilai rata-rata dari `Salary`
var(employee$salary)                    # Temukan nilai variansi dari `Salary`
sd(employee$salary)                     # Temukan nilai standar deviasi dari `Salary`
summary(employee)                       # Ringkasan statistik sederhana dari `employee`
```

Catatan: Berhati-hatilah saat mengekstrak kumpulan data yang berisi nilai yang hilang, jangan lupa mengabaikannya, atau menghapusnya terlebih dahulu. Lihat contoh berikut:

```
#View(mtcars)                          # Lihat dataset `mtcars` (environment R)
#?mtcars                               # Informasi detail tentang mtcars
min(mtcars$mpg )                       # Temukan minimum gallon Miles/(US)
```

```
## [1] 10.4
```

```
max(mtcars$mpg , na.rm = TRUE)         # Temukan maksimum gallon Miles/(US)
```

```
## [1] 33.9
```

```
mean(mtcars$mpg , na.rm = TRUE)        # Temukan rata-rata gallon Miles/(US)
```

```
## [1] 20.09
```

```
var(mtcars$mpg , na.rm = TRUE)         # Temukan varians gallon Miles/(US)
```

```
## [1] 36.32
```

```
sd(mtcars$mpg , na.rm = TRUE)          # Temukan standar deviasi gallon Miles/(US)
```

```
## [1] 6.027
```

4.6 Mengubah Nama

Objek R dapat memiliki nama, yang sangat berguna untuk menulis kode yang dapat dibaca dan menggambarkan objek sendiri. Di sini, Anda akan belajar cara mengganti nama kolom dari dataframe pada R menggunakan fungsi `names()`.

```
rename_1<-employee[,1:5]                # Mengubah nama variabel
names(rename_1)<-c("gaji",
                   "usia",
                   "gaji_awal",
                   "jam_kerja",
                   "pengalaman_kerja")# Mengubah nama variabel menjadi Indonesia
rename_1                                # Cetak hasilnya
```

Jika dataframe Anda dihasilkan dari fungsi `matrix()`, Anda dapat mengubah nama kolom dan barisnya.

```
m <- matrix(1:52, nrow = 26, ncol = 2)
dimnames(m) <- list(c(LETTERS), c("AA", "BB"))
m
```

```
##   AA BB
## A  1 27
## B  2 28
## C  3 29
## D  4 30
## E  5 31
## F  6 32
## G  7 33
## H  8 34
## I  9 35
## J 10 36
## K 11 37
## L 12 38
## M 13 39
## N 14 40
## O 15 41
## P 16 42
## Q 17 43
## R 18 44
## S 19 45
## T 20 46
## U 21 47
## V 22 48
## W 23 49
## X 24 50
## Y 25 51
## Z 26 52
```

Nama kolom dan nama baris dapat diatur secara terpisah menggunakan fungsi `colnames()` dan `rownames()`.

```r
colnames(m) <- c("Column 1", "Column 2") # Mengubah nama kolom
rownames(m) <- c(letters)                # Mengubah nama baris
m
```

```
##   Column 1 Column 2
## a        1       27
## b        2       28
## c        3       29
## d        4       30
## e        5       31
## f        6       32
## g        7       33
## h        8       34
## i        9       35
## j       10       36
## k       11       37
## l       12       38
## m       13       39
## n       14       40
## o       15       41
## p       16       42
## q       17       43
## r       18       44
## s       19       45
## t       20       46
## u       21       47
## v       22       48
## w       23       49
## x       24       50
## y       25       51
## z       26       52
```

4.7 Latihan

Gunakan berikut ini:

```r
data_frame <- fake_products(100)         # Rekayasa Data Produk
View(data_frame)
```

1. Lakukan Proses ekstraksi data seperti materi diatas!
2. Hintunglah Jumlah produk yang berasal dari suatu Negara (Contoh: “Taiwan”)!
3. Hitunglah Jumalah Produk yang ada dalam dataset tersebut!

Catatan: Kumpulkan hasil latihan anda, tidak boleh sama dengan teman mahasiwa lainnya.

Chapter 5

Fungsi dalam R

Suatu fungsi *(function)* dalam lingkungan pemrograman adalah satu set instruksi untuk melaksanakan tugas-tugas tertentu. Seorang programmer membangun sebuah fungsi untuk menghindari pengulangan tugas yang sama atau mengurangi kompleksitas.

Dari gambar diata diperlihatkan bahwa:

- Jika $x = 5$ adalah masukan *(input)*
- Diberikan instruksi (fungsi) $f(x) = x + 3$, yang disebut badan progam *(body)*
- Maka $y = 8$ aadalah keluaran *(output)*

Sehingga, komponen yang harus terkandung didalam fungsi adalah:

- f adalah `Nama Fungsi` yang digunakan untuk menjalan fungsi (perintah) pada program tertentu.
- x adalah `Masukan`, tetapi mungkin saja tidak ada argumen.
- $x + 3$ adalah `Badan Program` yang mendefinisikan fungsi yang dilakukan.
- `Keluaran` adalah perintah pengembalian satu atau lebih nilai dan mungkin saja tidak memuat pengembalian nilai.

Dalam berbagai kesempatan, programer diharapkan untuk mampu membangun fungsi (algoritma) sendiri dikarenakan tugas tertentu tidak dapat diselesaikan dengan fungsi yang sudah ada atau tidak ditemukannya fungsi dalam bahasa pemoragraman yang sedang digunakan. Pembentukan fungsi ini akan dibagi menjadi dua, yaitu fungsi dengan satu-argumen dan fungsi dengan multi-argument.

5.1 Fungsi Satu Argumen

Fungsi menerima nilai dan mengembalikan kuadrat dari suatu nilai. Berikut ini, dilampirkan struktur penulisan suatu fungsi dengan satu argumen secara garis besar:

```
nama.fungsi <- function(argumen)
  {
     perhitungan perintah yang dilakukan terhadap argumen
     beberapa kode lain
  }
```

5.1.1 Fungsi Kuadrat

Jika diberikan suatu vector $x = 1, 2, 3, \cdots, n$, maka nilai x^2 dapat dihitung dengan fungsi berikut;

```
masukan=c(1:5)                                  # nilai x awal (1,2,3,4,5)

x_kuadrat <- function(x)                        # nama fungsi dan argumen
   {                                            # pembukaan fungsi
    x*x                                         # perintah yang dilakukan
   }                                            # penutupan fungsi

x_kuadrat(masukan)                              # menggunakan fungsi
```

```
## [1]  1  4  9 16 25
```

```
cat("Hasil kuadrat:",x_kuadrat(masukan))        # menggunakan fungsi
```

```
## Hasil kuadrat: 1 4 9 16 25
```

atau,

```
masukan=c(1:5)                                  # nilai x awal (1,2,3,4,5)

x_kuadrat <- function(x)                        # nama fungsi dan argumen
   {                                            # pembukaan fungsi
    keluaran = x^2                              # perintah yang dilakukan
    return(cat("Hasil kuadrat:",keluaran))      # print hasil dengan komentar
   }                                            # penutupan fungsi

x_kuadrat(masukan)                              # menggunakan fungsi
```

```
## Hasil kuadrat: 1 4 9 16 25
```

5.1.2 Fungsi Akar

Jika diberikan suatu vector adalah kelipatan 9 dari 3 sampai 27, maka nilai $\sqrt[3]{x}$ dapat dihitung dengan fungsi berikut;

```
masukan = seq(from=3, to=27, by = 9)          # kelipatan 9 dari 3 sampai 27

akar <- function(x)                           # nama fungsi dan argumen
   {                                          # pembukaan fungsi
    x^(1/3)                                   # perintah yang dilakukan
   }                                          # penutupan fungsi

akar(masukan)                                 # menggunakan fungsi
```

```
## [1] 1.442 2.289 2.759
```

5.1.3 Nilai Rerata

Jika diberikan suatu vector adalah 100 sampel acak dengan rata-rata 160 dan standar deviasi 15, maka nilai rata-ratanya $\bar{x}$ dapat dihitung dengan fungsi berikut;

```
masukan = rnorm(100, mean=160, sd=15)         # 100 sampel,mean=160, dan sd=15

rerata<- function(x)                          # nama fungsi dan argumen
{                                             # pembukaan fungsi
  sum(x)/length(x)                            # perintah yang dilakukan
}                                             # penutupan fungsi

rerata(masukan)                               # menggunakan fungsi
```

```
## [1] 156.5
```

5.1.4 Konversi Persen

Misalkan anda ingin menyajikan angka pecahan sebagai nilai persentase, dibulatkan dengan baik ke dalam dua digit desimal. Berikut cara mencapainya:

- Kalikan bilangan pecahan dengan 100.
- Bulatkan hasilnya ke satu tempat desimal: Anda dapat menggunakan fungsi `round()` untuk melakukan ini.
- Tempelkan tanda persentase setelah angka yang dibulatkan dengan menggunakan fungsi `paste()`

- Cetak hasilnya: dengan menggunakan `print()`.

Sebenarnya, akan sangat mudah menerjemahkan langkah-langkah ini ke dalam skrip R berikut ini:

```
x <- c(0.8765, 0.4321, 0.1234, 0.05678)
persen <- round(x*100, digits = 2)
a <- paste(persen, "%")
print(a)
```

```
## [1] "87.65 %" "43.21 %" "12.34 %" "5.68 %"
```

Untuk membuat skrip ini menjadi sebuah fungsi, Anda perlu melakukan beberapa hal berikut:

```
x <- c(0.8765, 0.4321, 0.1234, 0.05678)

persen <- function(x)
  {
    persen_2digit <- round(x * 100, digits = 2)
    keluaran <- paste(persen_2digit, "%")
     return(keluaran)
}

persen(x)
```

```
## [1] "87.65 %" "43.21 %" "12.34 %" "5.68 %"
```

5.2 Simpan Fungsi

Jika anda menyimpan skrip ini sebagai file `.R`: misalnya, `percent.R` ke komputer/PC anda dalam sebuah folder. Kemudian, kapapun anda dapat memanggil skrip ini di konsol dengan perintah berikut:

```
x <- c(0.8765, 0.4321, 0.1234, 0.05678)
source('persen.R')                    # pastikan direktori anda benar
persen(x)                                    # menggunakan fungsi anda
```

```
## [1] "87.65 %" "43.21 %" "12.34 %" "5.68 %"
```

Catatan: Sebenarnya sudah ada library yang dapat anda gunakan untuk mengubah suatu array di R yaitu:

```
library(scales)
percent(x,accuracy=0.01)
```

```
## [1] "87.65%" "43.21%" "12.34%" "5.68%"
```

5.3 Fungsi Multi Argumen

Kita dapat menulis fungsi dengan lebih dari satu argumen. Berikut Ini adalah fungsi langsung mengalikan dua variabel.

```
nama.fungsi <- function(argumen1, argumen2,...., argumen_n)
    {
     perhitungan perintah yang dilakukan terhadap argumen
     beberapa kode lain
    }
```

5.3.1 Luas & Keliling

Misalkan anda ingin menghitung luas dan keliling suatu persegi (panjang), diketahui panjang p dan lebarnya l. Berikut ini diperlihatkan penyelesaian dengan menggunakan fungsi:

```
luas_keliling <- function(p,l)                  # nama fungsi dan argumen
{                                               # pembukaan fungsi
  luas= p*l                                     # menghitung luas
  keliling= 2*(p+l)                             # menghitung keliling
  return (cat(c("Luas:",luas, "\n",
              "Keliling:",keliling)))
}                                               # penutupan fungsi
luas_keliling(8,6)                              # menggunakan fungsi
```

```
## Luas: 48
##  Keliling: 28
```

5.3.2 Luas, Keliling dan Volume

Misalkan anda ingin menghitung luas, keliling dan volume suatu balok (kubus), diketahui panjang p, lebarnya l, dan tinggi t. Berikut ini diperlihatkan penyelesaian dengan menggunakan fungsi:

```
lukelvol <- function(p,l,t)                    # nama fungsi dan argumen
{                                              # pembukaan fungsi
  luas_permukaan = 2*((p*l)+(p*l)+(t*l))       # menghitung luas permukan
  keliling = 2*(p +l)                          # menghitung keliling rusuk
  volume = p*l*t                               # menghitung volume balok (kubus)
  return (cat(c("Luas Permukaan:",
                luas_permukaan, sep = "\n",
                "Keliling:",keliling,sep="\n",
                "Volume:",volume)))
}                                              # penutupan fungsi
lukelvol(6,7,8)                                # menggunakan fungsi
```

```
## Luas Permukaan: 280
##  Keliling: 26
##  Volume: 336
```

5.3.3 Rerata Frekuensi

Jika diberikan data berfrekensi sebagai berikut:

Tinggi Badan	Frekuensi
150	16
155	23
160	28
165	40
170	39
175	22
180	9

Maka cara menghitung reratanya dengan bantuan fungsi adalah sebagai berikut:

```
Tinggi<-seq(150,180,5)                         # masukan/argumen 1
Frek<-c(15,23,28,40,39,22,9)                   # masukan/argumen 2

rerata_frek <- function(x,frek)                # nama fungsi dan argumen
{                                              # pembukaan fungsi
 keluaran=sum(x*frek)/length(frek)             # menghitung rerata frekuensinya
 return(cat("Reratanya:",keluaran))            # print hasil dengan komentar
}                                              # penutupan fungsi

rerata_frek(Tinggi,Frek)                       # menggunakan fungsi
```

```
## Reratanya: 4142
```

5.4 Fungsi untuk Data Frame

5.4.1 Normalisasi

Seperti yang telah saya sebutkan sebelumnya, ilmuwan data perlu melakukan banyak tugas berulang. Sebagian besar waktu, kami menyalin dan menempelkan potongan kode berulang-ulang. Contoh lain, normalisasi suatu variabel sangat disarankan sebelum kita menjalankan algoritma pembelajaran mesin. Rumus untuk menormalkan variabel adalah:

$$\text{Normalisasi} = x - x_{min}/x_{max} - x_{min}$$

Mari kita buat kerangka data seperti yang telah kita pelajari di dasar-dasar R di bagian terakhir.

```
set.seed(123)                                   # statik random set
a = rnorm(100, 5, 1)                            # 100 bilangan acak normal
b = rnorm(100, 5, 1)                            # 100 bilangan acak normal
c = rnorm(100, 5, 1)                            # 100 bilangan acak normal
df<- data.frame(a,b,c)                          # buat data frame
df                                              # mencetak hasil data frame
```

```
##         a     b     c
## 1   4.440 4.290 7.199
## 2   4.770 5.257 6.312
## 3   6.559 4.753 4.735
## 4   5.071 4.652 5.543
## 5   5.129 4.048 4.586
## 6   6.715 4.955 4.524
## 7   5.461 4.215 4.211
## 8   3.735 3.332 4.405
## 9   4.313 4.620 6.651
## 10  4.554 5.919 4.946
## 11  6.224 4.425 5.119
## 12  5.360 5.608 5.244
## 13  5.401 3.382 6.232
## 14  5.111 4.944 4.484
## 15  4.444 5.519 4.007
## 16  6.787 5.301 6.676
## 17  5.498 5.106 4.559
## 18  3.033 4.359 4.277
## 19  5.701 4.150 3.764
## 20  4.527 3.976 3.715
## 21  3.932 5.118 4.426
```

```
## 22  4.782 4.053 5.618
## 23  3.974 4.509 6.110
## 24  4.271 4.744 5.708
## 25  4.375 6.844 4.636
## 26  3.313 4.348 5.060
## 27  5.838 5.235 4.295
## 28  5.153 5.078 4.283
## 29  3.862 4.038 5.885
## 30  6.254 4.929 3.984
## 31  5.426 6.445 6.955
## 32  4.705 5.452 4.910
## 33  5.895 5.041 5.215
## 34  5.878 4.578 4.261
## 35  5.822 2.947 4.426
## 36  5.689 6.131 3.683
## 37  5.554 3.539 4.817
## 38  4.938 5.740 5.419
## 39  4.694 6.909 5.324
## 40  4.620 3.556 4.218
## 41  4.305 5.702 4.211
## 42  4.792 4.738 4.498
## 43  3.735 3.428 6.496
## 44  7.169 3.485 3.863
## 45  6.208 3.398 4.821
## 46  3.877 4.469 6.902
## 47  4.597 3.538 4.899
## 48  4.533 5.688 3.640
## 49  5.780 7.100 4.335
## 50  4.917 3.713 5.485
## 51  5.253 5.788 4.624
## 52  4.971 5.769 4.438
## 53  4.957 5.332 4.656
## 54  6.369 3.992 5.090
## 55  4.774 4.881 6.599
## 56  6.516 4.720 4.911
## 57  3.451 5.563 6.081
## 58  5.585 4.628 5.631
## 59  5.124 5.977 4.886
## 60  5.216 4.625 3.467
## 61  5.380 6.053 4.479
## 62  4.498 3.951 4.510
## 63  4.667 3.740 5.047
## 64  3.981 8.241 6.300
## 65  3.928 4.583 7.293
## 66  5.304 5.298 6.548
## 67  5.448 5.637 4.867
```

```
## 68  5.053 4.516 3.243
## 69  5.922 5.517 4.611
## 70  7.050 5.369 5.089
## 71  4.509 4.785 5.845
## 72  2.691 5.065 5.963
## 73  6.006 4.966 5.684
## 74  4.291 7.128 3.605
## 75  4.312 4.259 5.850
## 76  6.026 3.904 4.553
## 77  4.715 5.038 5.175
## 78  3.779 5.310 5.075
## 79  5.181 5.437 5.428
## 80  4.861 4.542 5.025
## 81  5.006 3.937 3.333
## 82  5.385 6.263 5.736
## 83  4.629 4.650 5.386
## 84  5.644 4.134 4.734
## 85  4.780 4.764 5.118
## 86  5.332 4.803 5.134
## 87  6.097 6.110 5.221
## 88  5.435 5.085 6.641
## 89  4.674 5.754 4.781
## 90  6.149 4.501 5.168
## 91  5.994 5.214 6.168
## 92  5.548 4.675 6.054
## 93  5.239 5.095 6.145
## 94  4.372 4.105 4.423
## 95  6.361 3.689 7.002
## 96  4.400 6.997 5.067
## 97  7.187 5.601 6.867
## 98  6.533 3.749 3.649
## 99  4.764 4.389 5.021
## 100 3.974 3.815 6.250
```

```
typeof(df)
```

```
## [1] "list"
```

Kita sudah mengetahui cara menggunakan fungsi `min()` dan `max()` di R. Oleh karena itu kita dapat menggunakan rumus normalisasi yang kita miliki di atas untuk mendapatkan nilai normalisasi `df` sebagai berikut:

```
df.norm <- data.frame(
  a = (df$a -min(df$a))/(max(df$a)-min(df$a)),
  b = (df$b -min(df$b))/(max(df$b)-min(df$b)),
```

```
  c = (df$c -min(df$c))/(max(df$c)-min(df$c))
)
df.norm
```

```
##           a       b       c
## 1   0.38890 0.25364 0.97672
## 2   0.46236 0.43634 0.75784
## 3   0.86020 0.34123 0.36828
## 4   0.52923 0.32218 0.56789
## 5   0.54230 0.20808 0.33144
## 6   0.89497 0.37932 0.31615
## 7   0.61605 0.23957 0.23902
## 8   0.23220 0.07278 0.28692
## 9   0.36080 0.31600 0.84142
## 10  0.41443 0.56141 0.42041
## 11  0.78578 0.27915 0.46320
## 12  0.59357 0.50266 0.49393
## 13  0.60268 0.08223 0.73810
## 14  0.53816 0.37733 0.30632
## 15  0.38993 0.48593 0.18867
## 16  0.91095 0.44471 0.84755
## 17  0.62427 0.40778 0.32481
## 18  0.07618 0.26680 0.25520
## 19  0.66953 0.22733 0.12847
## 20  0.40840 0.19438 0.11651
## 21  0.27607 0.41004 0.29202
## 22  0.46507 0.20886 0.58636
## 23  0.28537 0.29517 0.70782
## 24  0.35145 0.33945 0.60848
## 25  0.37454 0.73610 0.34395
## 26  0.13844 0.26468 0.44851
## 27  0.69987 0.43228 0.25976
## 28  0.54766 0.40255 0.25664
## 29  0.26043 0.20614 0.65221
## 30  0.79239 0.37435 0.18296
## 31  0.60839 0.66067 0.91659
## 32  0.44793 0.47310 0.41145
## 33  0.71262 0.39561 0.48673
## 34  0.70884 0.30802 0.25138
## 35  0.69626 0.00000 0.29191
## 36  0.66670 0.60151 0.10853
## 37  0.63674 0.11193 0.38858
## 38  0.49978 0.52759 0.53722
## 39  0.44550 0.74842 0.51384
## 40  0.42893 0.11510 0.24076
```

```
## 41  0.35905 0.52038 0.23901
## 42  0.46731 0.33830 0.30974
## 43  0.23213 0.09087 0.80319
## 44  0.99591 0.10173 0.15291
## 45  0.78219 0.08532 0.38954
## 46  0.26377 0.28754 0.90352
## 47  0.42395 0.11172 0.40882
## 48  0.40977 0.51776 0.09796
## 49  0.68701 0.78450 0.26960
## 50  0.49501 0.14473 0.55363
## 51  0.56988 0.53661 0.34100
## 52  0.50720 0.53308 0.29500
## 53  0.50401 0.45057 0.34883
## 54  0.81792 0.19736 0.45610
## 55  0.46334 0.36526 0.82848
## 56  0.85080 0.33486 0.41188
## 57  0.16911 0.49416 0.70064
## 58  0.64356 0.31748 0.58951
## 59  0.54109 0.57236 0.40569
## 60  0.56157 0.31707 0.05522
## 61  0.59798 0.58666 0.30507
## 62  0.40183 0.18965 0.31279
## 63  0.43944 0.14980 0.44540
## 64  0.28702 1.00000 0.75482
## 65  0.27519 0.30909 1.00000
## 66  0.58105 0.44415 0.81591
## 67  0.61323 0.50806 0.40087
## 68  0.52534 0.29645 0.00000
## 69  0.71866 0.48545 0.33775
## 70  0.96948 0.45751 0.45578
## 71  0.40434 0.34714 0.64242
## 72  0.00000 0.40016 0.67144
## 73  0.73722 0.38139 0.60273
## 74  0.35583 0.78985 0.08921
## 75  0.36054 0.24780 0.64356
## 76  0.74163 0.18081 0.32348
## 77  0.45022 0.39496 0.47692
## 78  0.24207 0.44647 0.45216
## 79  0.55387 0.47027 0.53948
## 80  0.48266 0.30125 0.43985
## 81  0.51483 0.18698 0.02199
## 82  0.59923 0.62642 0.61562
## 83  0.43111 0.32178 0.52908
## 84  0.65685 0.22434 0.36815
## 85  0.46451 0.34319 0.46293
## 86  0.58733 0.35058 0.46685
```

```
## 87  0.75748 0.59747 0.48833
## 88  0.61033 0.40383 0.83894
## 89  0.44106 0.53025 0.37966
## 90  0.76904 0.29352 0.47525
## 91  0.73450 0.42833 0.72227
## 92  0.63551 0.32650 0.69407
## 93  0.56664 0.40569 0.71656
## 94  0.37390 0.21870 0.29115
## 95  0.81615 0.14024 0.92824
## 96  0.38005 0.76506 0.45022
## 97  1.00000 0.50129 0.89475
## 98  0.85439 0.15148 0.10016
## 99  0.46113 0.27238 0.43893
## 100 0.28528 0.16391 0.74240
```

Namun, metode ini rentan terhadap kesalahan penulisan koding. Kita bisa menyalin koding yang hampir serupa, tetapi mungkin saja lupa mengubah variabel yang perlu diganti. Oleh karena itu sebaiknya, kita dapat pertimbangkan penggunaan suatu fungsi untuk melakukannya:

```
normalize <- function(x){
  norm <- (x-min(x))/(max(x)-min(x))
  return(norm)
}
```

```
df.norm = normalize(df)
df.norm
```

```
##           a       b       c
## 1   0.31507 0.28805 0.81222
## 2   0.37458 0.46233 0.65251
## 3   0.69689 0.37160 0.36828
## 4   0.42875 0.35343 0.51392
## 5   0.43935 0.24459 0.34140
## 6   0.72506 0.40794 0.33024
## 7   0.49910 0.27463 0.27397
## 8   0.18812 0.11553 0.30892
## 9   0.29230 0.34754 0.71350
## 10  0.33575 0.58163 0.40632
## 11  0.63660 0.31239 0.43754
## 12  0.48088 0.52559 0.45996
## 13  0.48826 0.12455 0.63811
## 14  0.43599 0.40604 0.32307
## 15  0.31590 0.50963 0.23723
## 16  0.73801 0.47031 0.71797
```

```
## 17  0.50575 0.43509 0.33656
## 18  0.06172 0.30061 0.28577
## 19  0.54242 0.26296 0.19331
## 20  0.33087 0.23153 0.18458
## 21  0.22366 0.43725 0.31264
## 22  0.37678 0.24534 0.52740
## 23  0.23119 0.32767 0.61602
## 24  0.28472 0.36991 0.54354
## 25  0.30344 0.74827 0.35053
## 26  0.11215 0.29859 0.42682
## 27  0.56700 0.45846 0.28910
## 28  0.44368 0.43010 0.28683
## 29  0.21099 0.24275 0.57544
## 30  0.64195 0.40320 0.23307
## 31  0.49289 0.67632 0.76834
## 32  0.36289 0.49740 0.39978
## 33  0.57733 0.42348 0.45471
## 34  0.57427 0.33993 0.28299
## 35  0.56408 0.04611 0.31256
## 36  0.54013 0.61989 0.17876
## 37  0.51585 0.15288 0.38309
## 38  0.40490 0.54937 0.49154
## 39  0.36092 0.76002 0.47448
## 40  0.34750 0.15590 0.27524
## 41  0.29088 0.54249 0.27396
## 42  0.37859 0.36881 0.32557
## 43  0.18806 0.13279 0.68560
## 44  0.80684 0.14315 0.21114
## 45  0.63369 0.12750 0.38379
## 46  0.21370 0.32040 0.75881
## 47  0.34346 0.15268 0.39786
## 48  0.33197 0.54000 0.17104
## 49  0.55658 0.79443 0.29628
## 50  0.40103 0.18416 0.50352
## 51  0.46169 0.55798 0.34838
## 52  0.41091 0.55461 0.31482
## 53  0.40833 0.47590 0.35409
## 54  0.66264 0.23437 0.43236
## 55  0.37537 0.39453 0.70406
## 56  0.68928 0.36553 0.40009
## 57  0.13701 0.51749 0.61078
## 58  0.52138 0.34895 0.52970
## 59  0.43837 0.59208 0.39558
## 60  0.45496 0.34856 0.13986
## 61  0.48445 0.60572 0.32216
## 62  0.32555 0.22702 0.32779
```

```
## 63  0.35602 0.18900 0.42455
## 64  0.23253 1.00000 0.65031
## 65  0.22294 0.34094 0.82920
## 66  0.47074 0.46978 0.69488
## 67  0.49681 0.53074 0.39206
## 68  0.42560 0.32889 0.09957
## 69  0.58222 0.50918 0.34600
## 70  0.78542 0.48253 0.43212
## 71  0.32758 0.37724 0.56830
## 72  0.00000 0.42781 0.58947
## 73  0.59726 0.40991 0.53935
## 74  0.28827 0.79954 0.16466
## 75  0.29209 0.28248 0.56913
## 76  0.60083 0.21858 0.33559
## 77  0.36474 0.42286 0.44755
## 78  0.19611 0.47199 0.42948
## 79  0.44872 0.49470 0.49320
## 80  0.39103 0.33347 0.42050
## 81  0.41709 0.22447 0.11562
## 82  0.48547 0.64364 0.54875
## 83  0.34927 0.35305 0.48560
## 84  0.53215 0.26011 0.36819
## 85  0.37632 0.37348 0.43734
## 86  0.47583 0.38052 0.44020
## 87  0.61367 0.61603 0.45587
## 88  0.49446 0.43132 0.71169
## 89  0.35733 0.55191 0.37658
## 90  0.62304 0.32609 0.44633
## 91  0.59505 0.45469 0.62656
## 92  0.51486 0.35755 0.60599
## 93  0.45906 0.43309 0.62240
## 94  0.30292 0.25473 0.31201
## 95  0.66120 0.17988 0.77684
## 96  0.30790 0.77590 0.42807
## 97  0.81015 0.52428 0.75241
## 98  0.69219 0.19060 0.17265
## 99  0.37358 0.30593 0.41983
## 100 0.23112 0.20246 0.64125
```

5.5 Latihan

1. Buatlah Fungsi Summary Statistik untuk data berfrekuansi
2. Buatlah Fungsi summary Statistik untuk data frame

Catatan: Kumpulkan hasil latihan anda, tidak boleh sama dengan teman mahasiwa lainnya.

Chapter 6

Struktur Kontrol

Pada dasarnya, struktur kontrol memungkinkan Anda untuk memasukkan beberapa "logika" ke dalam kode R Anda, daripada hanya mengeksekusi kode R yang sama setiap saat. Struktur kontrol memungkinkan Anda untuk merespons input atau fitur data dan mengeksekusi ekspresi R yang berbeda. Struktur kontrol yang umum digunakan adalah

- `if` dan `else` menguji suatu kondisi dan menindaklanjutinya
- `for` mengeksekusi loop beberapa kali
- `while` menjalankan loop saat kondisi benar
- `break` mengeksekusi fraagmen loop yang rusak
- `repeat` menjalankan loop tak terbatas (harus keluar dari itu untuk berhenti)
- `next` melewati interasi dari sebuah loop

Sebagian besar struktur kontrol tidak digunakan dalam sesi interaktif, melainkan saat menulis fungsi atau ekspresi yang lebih panjang. Namun, konstruksi ini tidak harus digunakan dalam fungsi dan ada baiknya Anda memahaminya sebelumnya.

6.1 Pengambilan Keputusan

Gambaran pengambilan Keputusan *(Decision Tree)* dalam dunia pemrograman adalah proses penentuan keputusan dengan pernyataan bersyarat. Dalam hal ini, programer menginstruksikan komputer untuk melakukan suatu aksi tertentu (X), hanya jika suatu kondisi Y terpenuhi.

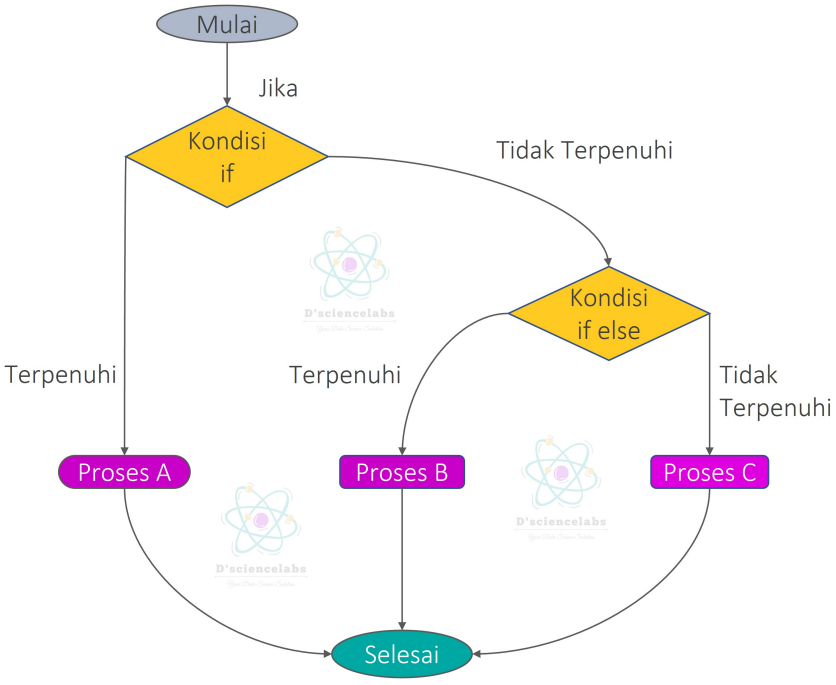

Figure 6.1: Struktur Kontrol if else

6.1.1 Sintaks `if`, `else-if`, dan `else`

Secara garis besar struktur kontrol pengambilan keputasan yang paling umum digunakan dalam bahasa pemrograman R adalah kombinasi `if-else`, bahkan untuk semua bahasa pemrograman lainnya. Struktur kontrol ini memungkinkan untuk menguji suatu kondisi dan menindaklanjutinya tergantung pada apakah itu benar atau salah.

Anda dapat memiliki serangkaian tes dengan mengikuti inisial `if` dengan sejumlah `else if` dan `else` itu sendiri. Fungsi umum untuk ini, seperti kode berikut:

```
nama.fungsi <- function(argumen){
  if (kondisi1) {
      lakukan sesuatu
      }
  else if (kondisi2) {
      lakukan sesuatu yang berbeda dibandingkan dengan kondisi 1}
  else if (kondisi3) {
      lakukan sesuatu yang berbeda dibandingkan dengan kondisi 1,2}
  else if (kondisi4) {
      lakukan sesuatu yang berbeda dibandingkan dengan kondisi 1,2,3}
  else (opsional){
      melakukan sesuatu yang berbeda dibandingkan semua kondisi semua}
}
```

6.1.2 Program Penilaian

Andaikan anda ingin membuat suatu program yang dapat memudahkan anda untuk memberikan peringkat kepada nilai mahasiswa sesuai dengan standar yang berlaku. Perhatikan contoh berikut dalam R:

```
x <- runif(1, 50, 100)                    # pilih satu nomor acak dari 0~100
nilai <- function(x){
            if (x>=85){                   # kondisi 1
              print('Peringkat A')}       # hasil untuk kondisi 1
            else if (x<85 & x >=70){      # kondisi 2
              print('Peringkat B')}       # hasil untuk kondisi 2
            else if (x<=69 & x >=60){     # kondisi 3
              print('Peringkat C')}       # hasil untuk kondisi 3
            else if (x<=59 & x >=50){     # kondisi 4
              print('Peringkat D')}       # hasil untuk kondisi 4
            else{                         # kondisi 5
              print('Gagal')}             # hasil untuk kondisi 5
```

```
}
nilai(x)
```

```
## [1] "Peringkat C"
```

6.1.3 Program Hitung Faktorial

Untuk menambah pemahaman terkait pengambilan keputusan dalam pemrograman, mari kita perhatiakan contoh berikut: fungsi factorial menghitung faktorial dari suatu bilangan n.

```
factorial <- function(n) {
  if (n == 0 || n == 1) {
    return(1)
  } else {
    return(n * factorial(n - 1))
  }
}

result <- factorial(5)
print(result)  # Output: 120
```

```
## [1] 120
```

Faktorial dari n (dilambangkan sebagai n!) adalah hasil perkalian semua bilangan bulat positif dari 1 hingga n. Dalam implementasi tersebut, fungsi factorial memanggil dirinya sendiri dengan argumen yang lebih kecil hingga mencapai kasus dasar (n = 0 atau n = 1), di mana hasil faktorialnya adalah 1.

Namun, perlu diingat bahwa pengulangan rekursif dapat menyebabkan pemanggilan berulang yang dalam beberapa kasus dapat mengakibatkan penumpukan memori (stack overflow) atau kinerja yang buruk. Oleh karena itu, perlu dipastikan bahwa ada kondisi berhenti yang jelas untuk menghindari pengulangan tak terbatas. Selain itu, dalam beberapa situasi, solusi non-rekursif atau pendekatan lain mungkin lebih efisien.

6.2 Pengulangan Rekursif

Pada saat melakukan analisis data, terkadang seorang data analis atau data scientist perlu menggunakan fungsi pengulangan dalam proses pembentukan, perhitungan, manipulasi struktur data seperti halnya vektor, matriks, list, data frame, atau objek lainnya.

6.2.1 Sintaks `for`

Dalam porose perulangan ini digunakan fungsi kontrol 'for', dimana setiap iterasi pada beberapa perintah akan dievaluasi melalui perulangan yang diinginkan.

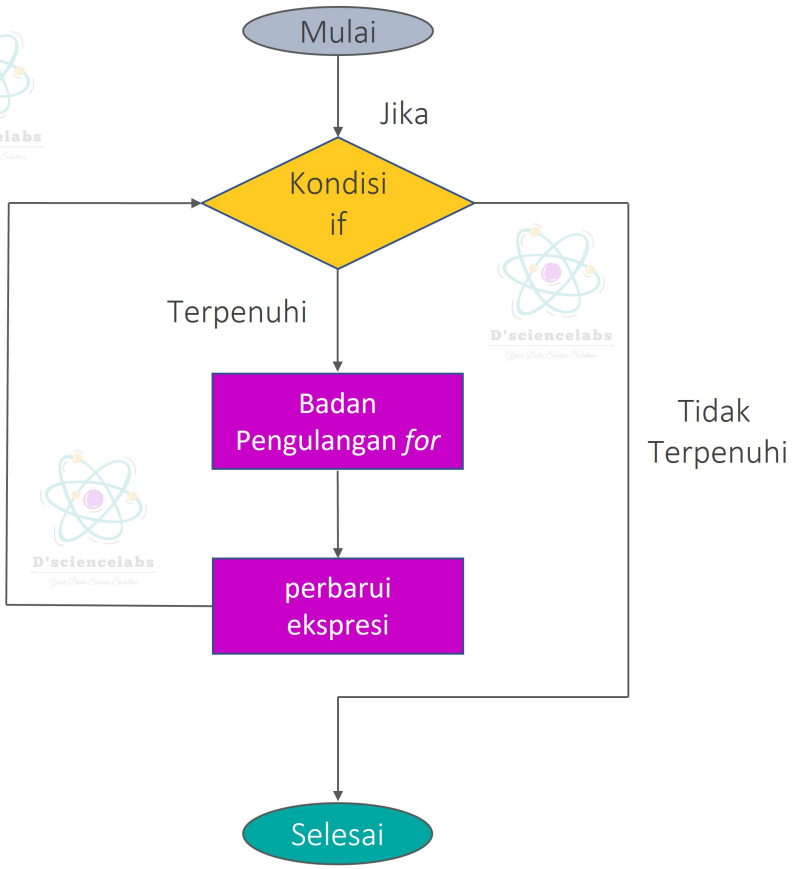

Figure 6.2: Struktur Kontrol for

Pada gambar di atas bisa dilihat bahwa perulangan juga memerlukan tes kondisi. Bila hasil tes kondisi terpenuhi, maka blok kode kembali dieksekusi. Tapi jika tidak terpenuhi, maka keluar dari perulangan. Sintaks pengulangan for di R sangat sederhana, dapat diperlihatkan sebagai berikut:

```
For (i in vector) {
    lakukan sesuatu
}
```

Jika anda ingin mencetak $i + 1$, menjadi $i = 1, \cdots 10$, pada setiap iterasi pengulangan (loop). Anda dapat menggunakan sintaks `for` untuk mencetak angka dengan yang dimulai $i = 0$ dan berakhir pada $i = 10$, sebagai berikut:

```
for (i in 0:10) {
  print(i + 1)
}
```

```
## [1] 1
## [1] 2
## [1] 3
## [1] 4
## [1] 5
## [1] 6
## [1] 7
## [1] 8
## [1] 9
## [1] 10
## [1] 11
```

Selain dapat digunakan pada data numerik, pengulangan for ini dapat juga diterapkan pada data karakter sebagai berikut:

```
mapel = c('Data Science', 'Statistika', 'Fisika','Algoritma dan Struktur Data')

for (i in mapel){
   print(paste0("Saya suka: ", i))
}
```

```
## [1] "Saya suka: Data Science"
## [1] "Saya suka: Statistika"
## [1] "Saya suka: Fisika"
## [1] "Saya suka: Algoritma dan Struktur Data"
```

Selain itu, anda juga dapat melakukan pengulangan bersarang untuk siklus tertentu. Sintaksnya direpresentasikan sebagai berikut:

```
nama = c("Bakti: ", "Alexander: ", "Siregar: ")
mapel = c('Data Science', 'Statistika','Algoritma dan Struktur Data')

for (x in nama)
  for (y in mapel){
    print(paste0(x, y))
  }
```

```
## [1] "Bakti: Data Science"
## [1] "Bakti: Statistika"
## [1] "Bakti: Algoritma dan Struktur Data"
## [1] "Alexander: Data Science"
## [1] "Alexander: Statistika"
## [1] "Alexander: Algoritma dan Struktur Data"
## [1] "Siregar: Data Science"
## [1] "Siregar: Statistika"
## [1] "Siregar: Algoritma dan Struktur Data"
```

6.2.2 Sintaks `while`

Terkadang anda bahkan tidak tahu berapa lama urutan input harus dijalankan. Hal ini biasa terjadi saat melakukan simulasi. Misalnya, Anda mungkin ingin mengulang sampai anda mendapatkan tiga kepala berturut-turut. Anda tidak dapat melakukan iterasi semacam itu dengan pengulangan `for`. Sebagai gantinya, Anda dapat menggunakan pengulangan `while`. Perulangan `while` lebih sederhana daripada perulangan for karena hanya memiliki dua komponen, kondisi, dan badan.

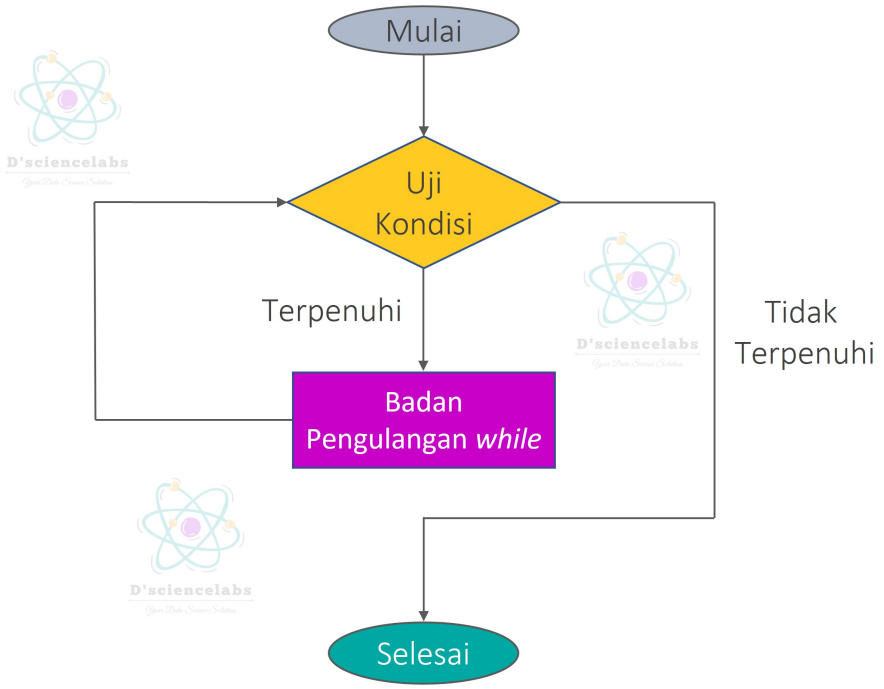

Figure 6.3: Struktur Kontrol while

Perulangan `while` dimulai dengan menguji suatu kondisi. Jika benar, maka mereka mengeksekusi badan perulangan. Setelah tubuh loop dieksekusi, kondisinya diuji lagi, dan seterusnya, sampai kondisinya salah, setelah itu loop akan melakukan eksekusi pada kodisi lainnya jika ada. Sintaks untuk `while` loop adalah sebagai berikut:

```
while (Kondisi) {
     lakukan sesuatu
}
```

Pengulangan while dalam bahasa pemrograman R digunakan untuk menjalankan serangkaian pernyataan selama kondisi tertentu tetap terpenuhi. Berikut adalah contoh penggunaan pengulangan while dalam R:

```
x <- 1

while (x <= 5) {
  print(x)
  x <- x + 1
}
```

```
## [1] 1
## [1] 2
## [1] 3
## [1] 4
## [1] 5
```

Dalam contoh di atas, loop while akan menjalankan blok pernyataan selama nilai x masih kurang dari atau sama dengan 5. Setiap kali loop dijalankan, nilai x akan ditambah 1, sehingga pada akhirnya, kondisi x <= 5 akan menjadi salah dan loop akan berhenti.

Anda perlu memastikan bahwa ada pernyataan di dalam blok while yang mengubah nilai yang digunakan dalam kondisi. Ini penting untuk menghindari terjebak dalam loop tak berakhir.

Berikut adalah contoh lain dengan menggunakan pengulangan while untuk menghitung jumlah bilangan genap antara 1 dan 10:

```
count <- 1
even_sum <- 0

while (count <= 10) {
  if (count %% 2 == 0) {
    even_sum <- even_sum + count
```

```
  }
  count <- count + 1
}

print(even_sum)  # Output: 30 (2 + 4 + 6 + 8 + 10)
```

```
## [1] 30
```

Dalam contoh ini, kita menggunakan loop while untuk menghitung jumlah bilangan genap antara 1 dan 10. Kita memeriksa apakah count (bilangan saat ini) adalah bilangan genap dengan menggunakan operator modulo (%%). Jika ya, kita menambahkan bilangan tersebut ke even_sum.

Penting untuk selalu memastikan bahwa kondisi dalam loop while pada suatu titik akan menjadi salah, sehingga loop akan berhenti dan program tidak terjebak dalam perulangan tak berakhir.

6.3 Interupsi Pengulangan

Interupsi Pengulangan dalam bahasa pemrograman dilakukan dengan pernyataan `break` pada suatu fungsi. Biasanya digunakan untuk melewati/menghentikan iterasi dan mengalirkan perintah pengulangan seperti (for, while, repeat). Meskipun, ini tidak umum digunakan dalam aplikasi statistik atau analisis data tetapi mereka memiliki peran penting dalam proses penyederhanaan program atau algoritma.

```
if (Test Kondisi) {
break
}
```

Andaikan kita ingin melakukan pengulangan suatu nilai pada vektor x, yang memiliki angka berurutan dari 1 hingga 5. Di dalam pengulangan `for`, kita telah menggunakan kondisi `if` untuk memutuskan pengulangan pada saat ditemukan nilai sama dengan 3. Dalam hal ini, loop akan berakhir ketika sesat pernyataan break terpenuhi.

```
x <- 1:100
for (val in x) {
if (val == 50){
break
}
print(val)
}
```

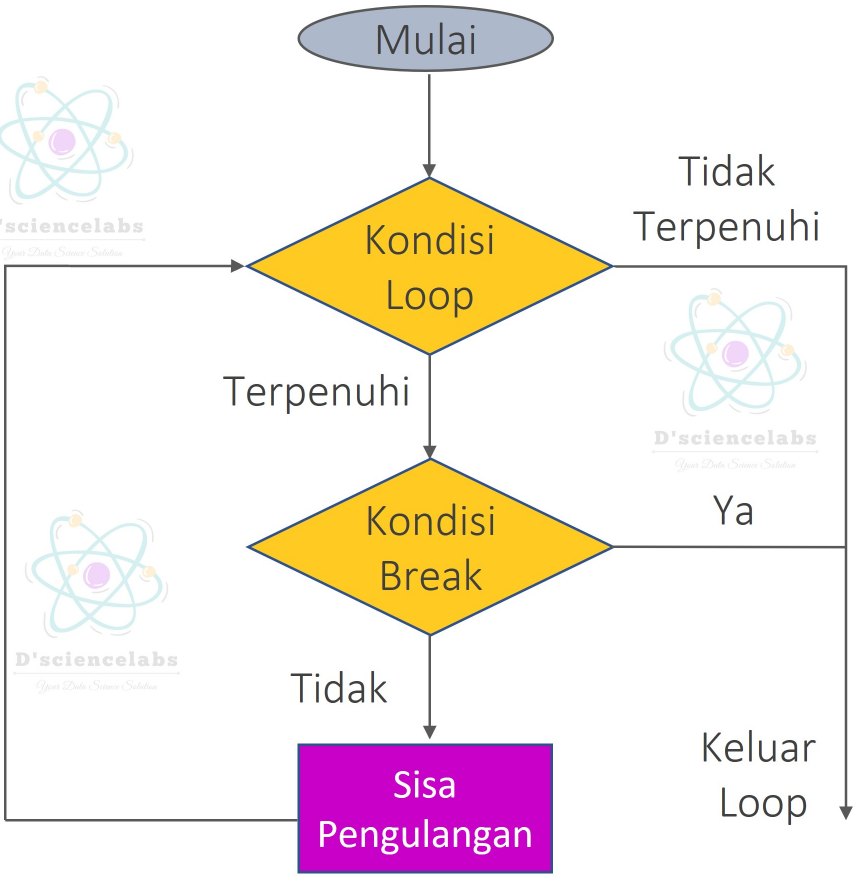

Figure 6.4: Struktur Kontrol break

```
## [1] 1
## [1] 2
## [1] 3
## [1] 4
## [1] 5
## [1] 6
## [1] 7
## [1] 8
## [1] 9
## [1] 10
## [1] 11
## [1] 12
## [1] 13
## [1] 14
## [1] 15
## [1] 16
## [1] 17
## [1] 18
## [1] 19
## [1] 20
## [1] 21
## [1] 22
## [1] 23
## [1] 24
## [1] 25
## [1] 26
## [1] 27
## [1] 28
## [1] 29
## [1] 30
## [1] 31
## [1] 32
## [1] 33
## [1] 34
## [1] 35
## [1] 36
## [1] 37
## [1] 38
## [1] 39
## [1] 40
## [1] 41
## [1] 42
## [1] 43
## [1] 44
## [1] 45
## [1] 46
```

```
## [1] 47
## [1] 48
## [1] 49
```

6.4 Pengulangan Berkala

Kondisi perulangan yang digunakan adalah `repeat` untuk mengulangi perintah beberapa kali. Tidak ada pemeriksaan kondisi di loop untuk keluar dalam fungsi pengulangan ini. Kita sendiri harus menempatkan kondisi secara eksplisit di dalam tubuh loop.

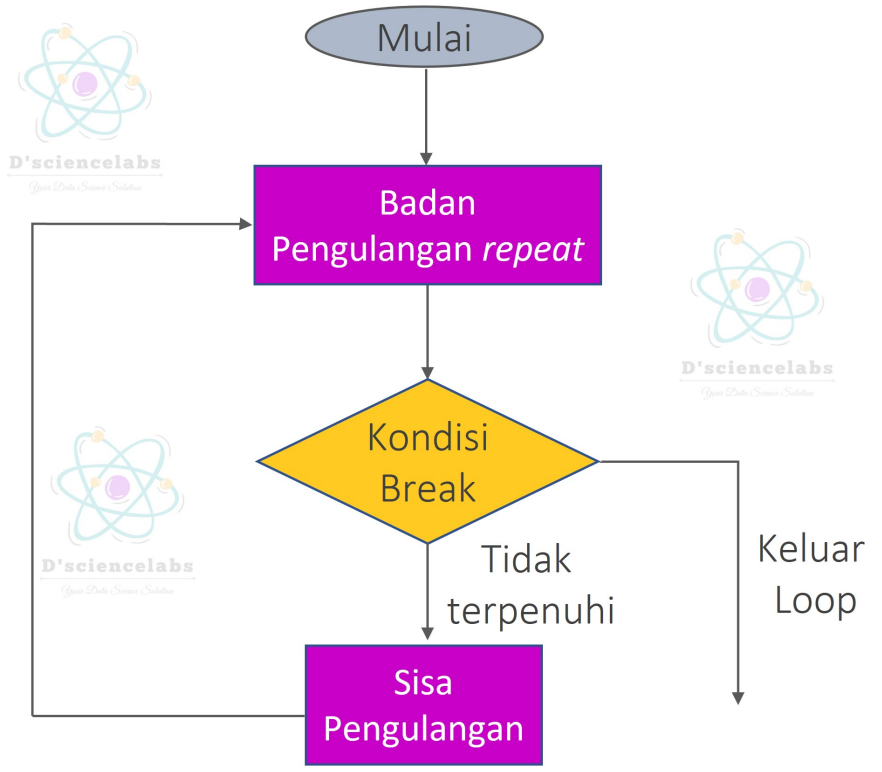

Figure 6.5: Struktur Kontrol repeat

Adapun sintak yang digunakan untuk melakukan pengulangan dengan repeat adalah sebgai berikut:

```
repeat {
lakukan sesuatu
}
```

Mari kita gunakan kondisi ini untuk memeriksa dan keluar dari loop ketika x mengambil nilai 6. Oleh karena itu, kita melihat dalam output bahwa hanya ada nilai dari 1 hingga 5 yang akan dicetak.

```
x <- 1
repeat {
  print(x)
    x = x+1
  if (x == 6){
  break
  }
}
```

```
## [1] 1
## [1] 2
## [1] 3
## [1] 4
## [1] 5
```

6.5 Skip Iterasi Pengulangan

Penyataan skip iterasi digunakan untuk melewati sisa kode di dalam satu lingkaran untuk iterasi saat ini saja. Artinya, tidak seperti pernyataan break, pengulangan tidak berhenti tetapi terus berlanjut dengan iterasi berikutnya.

```
for pengulangan {
  if (kondisi){
  next
  }
  print(val)
}
```

Gunakan pernyataan `next`, misalnya untuk memeriksa apakah nilainya sama dengan 5. Jika nilainya sama dengan 5, evaluasi saat ini berhenti (nilai tidak dicetak) tetapi perulangan berlanjut dengan iterasi berikutnya.

```
x <- 1:11
 for (val in x) {
 if (val == 5){
 next
 }
print(val)
}
```

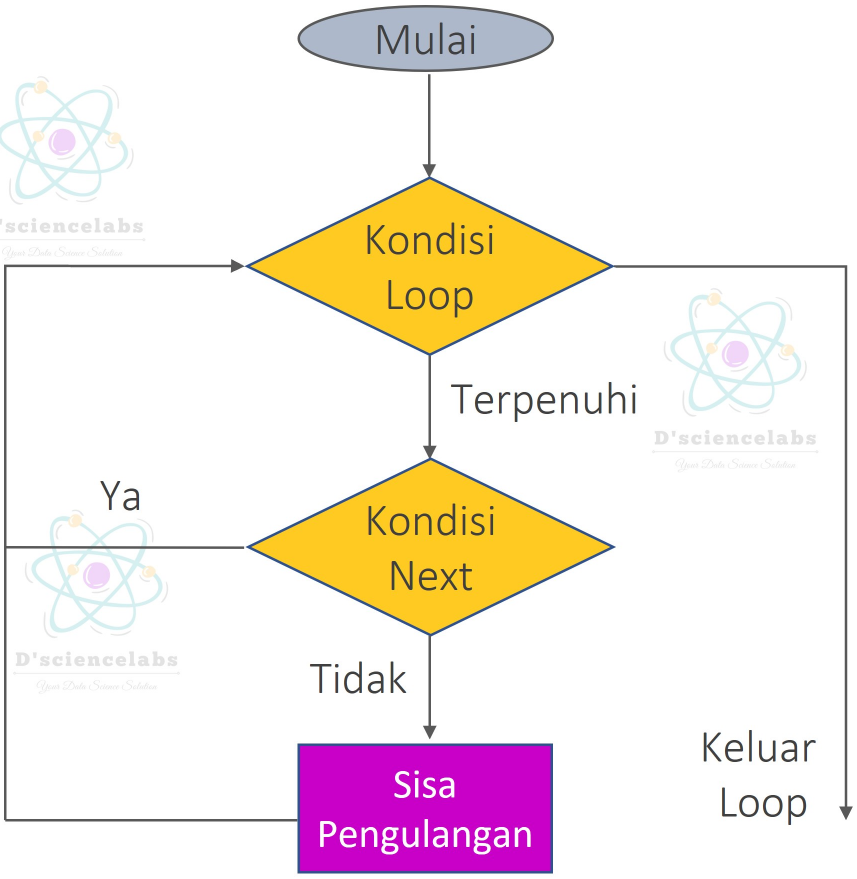

Figure 6.6: Struktur Kontrol next

```
## [1] 1
## [1] 2
## [1] 3
## [1] 4
## [1] 6
## [1] 7
## [1] 8
## [1] 9
## [1] 10
## [1] 11
```

6.6 Latihan

1. Pernyataan Kondisional (if-else): Buatlah sebuah program yang memeriksa apakah suatu bilangan adalah positif, negatif, atau nol, dan mencetak pesan yang sesuai.

2. Pengulangan (for): Hitunglah jumlah dari semua bilangan bulat genap antara 1 dan 50 menggunakan loop for.

3. Pengulangan (while): Buatlah program yang meminta pengguna untuk menebak suatu angka acak antara 1 dan 100. Berikan petunjuk apakah angka yang ditebak terlalu besar atau terlalu kecil, dan berhenti ketika pengguna berhasil menebak angka tersebut.

4. Pernyataan Switch: Buatlah program yang mengonversi nama hari dalam bahasa Inggris menjadi nama hari dalam bahasa Indonesia menggunakan pernyataan switch.

5. Pengulangan dan Pernyataan Kondisional Gabungan: Hitunglah jumlah bilangan bulat positif yang dapat dibagi habis oleh 3 atau 5 di antara 1 dan 1000, lalu cetak hasilnya.

6. Nested Loop: Buatlah pola segitiga angka seperti berikut menggunakan nested loop:

```
1
12
123
1234
12345
```

7. Penggunaan break: Buatlah sebuah program yang mencari bilangan prima pertama yang lebih dari 100.

8. Fungsi Rekursif: Buatlah fungsi rekursif untuk menghitung bilangan Fibonacci ke-n.

Catatan: Kumpulkan hasil latihan anda, tidak boleh sama dengan teman mahasiwa lainnya.

Chapter 7

Referensi

Berikut adalah beberapa referensi yang dapat Anda gunakan untuk mempelajari dasar-dasar pemrograman dalam bahasa R:

1. Venables, W.N. Smith D.M. and R Core Team. 2018. **An Introduction to R**: https://cran.r-project.org/manuals.html
2. R for Data Science: https://r4ds.had.co.nz/
3. Codecademy - Learn R : https://www.codecademy.com/learn/learn-r
4. DataCamp: https://www.datacamp.com/courses/tech:r
5. Primartha, R. 2018. **Belajar Machine Learning Teori dan Praktik**. Penerbit Informatika : Bandung
6. Rosadi,D. 2016. **Analisis Statistika dengan R**. Gadjah Mada University Press: Yogyakarta
7. STHDA. Running RStudio and Setting Up Your Working Directory - Easy R Programming .http://www.sthda.com/english/wiki/running-rstudio-and-setting-up-your-working-directory-easy-r-programming#set-your-working-directory
8. STDHA. **Getting Help With Functions In R Programming**. http://www.sthda.com/english/wiki/getting-help-with-functions-in-r-programming .